

ENADE

COMENTADO

COMPUTAÇÃO

2011

MARCELO HIDEKI YAMAGUTI
ANA PAULA TERRA BACELO
(Organizadores)



ENADE
COMENTADO

COMPUTAÇÃO
2011



Pontifícia Universidade Católica
do Rio Grande do Sul

CHANCELER

Dom Jaime Spengler

REITOR

Joaquim Clotet

VICE-REITOR

Evilázio Teixeira

CONSELHO EDITORIAL

Jorge Luis Nicolas Audy | **PRESIDENTE**

Jeronimo Carlos Santos Braga | **DIRETOR**

Jorge Campos da Costa | **EDITOR-CHEFE**

Agemir Bavaresco

Ana Maria Mello

Augusto Buchweitz

Augusto Mussi

Bettina Steren dos Santos

Carlos Gerbase

Carlos Graeff Teixeira

Clarice Beatriz da Costa Sohngen

Cláudio Luís C. Frankenberg

Erico João Hammes

Gilberto Keller de Andrade

Lauro Kopper Filho

ENADE

COMENTADO

COMPUTAÇÃO

2011

MARCELO HIDEKI YAMAGUTI
ANA PAULA TERRA BACELO
(Organizadores)



© EDIPUCRS, 2014

DESIGN GRÁFICO [CAPA] Rodrigo Braga

DESIGN GRÁFICO [DIAGRAMAÇÃO] Rodrigo Valls

REVISÃO DE TEXTO Fernanda Lisboa de Siqueira

Edição revisada segundo o novo Acordo Ortográfico da Língua Portuguesa.



EDIPUCRS – Editora Universitária da PUCRS

Av. Ipiranga, 6681 – Prédio 33
Caixa Postal 1429 – CEP 90619-900
Porto Alegre – RS – Brasil
Fone/fax: (51) 3320 3711
E-mail: edipucrs@pucrs.br – www.pucrs.br/edipucrs

Dados Internacionais de Catalogação na Publicação (CIP)

E56 ENADE comentado : computação 2011 [recurso eletrônico] /
orgs. Marcelo Hideki Yamaguti, Ana Paula Terra Bacelo. –
Dados eletrônicos. – Porto Alegre : EDIPUCRS, 2014.
80 p.

Sistema requerido: Adobe Acrobat Reader
Modo de acesso: <<http://www.pucrs.br/edipucrs>>
ISBN 978-85-397-0521-4

1. Educação Superior – Brasil – Avaliação. 2. Exame
Nacional de Cursos (Computação). 3. Computação – Ensino
Superior. I. Yamaguti, Marcelo Hideki. II. Bacelo, Ana Paula
Terra.

CDD 378.81

Ficha catalográfica elaborada pelo Setor de Tratamento da Informação da BC-PUCRS.

TODOS OS DIREITOS RESERVADOS. Proibida a reprodução total ou parcial, por qualquer meio ou processo, especialmente por sistemas gráficos, microfilmicos, fotográficos, reprográficos, fonográficos, videográficos. Vedada a memorização e/ou a recuperação total ou parcial, bem como a inclusão de qualquer parte desta obra em qualquer sistema de processamento de dados. Essas proibições aplicam-se também às características gráficas da obra e à sua editoração. A violação dos direitos autorais é punível como crime (art. 184 e parágrafos, do Código Penal), com pena de prisão e multa, conjuntamente com busca e apreensão e indenizações diversas (arts. 101 a 110 da Lei 9.610, de 19.02.1998, Lei dos Direitos Autorais).

Sumário

APRESENTAÇÃO	6	QUESTÃO 31	60
QUESTÃO 09	8	QUESTÃO 32	62
QUESTÃO 10	11	QUESTÃO 33	64
QUESTÃO 11	13	QUESTÃO 34	66
QUESTÃO 12	15	QUESTÃO 35	68
QUESTÃO 13 (ANULADA)	17	QUESTÃO 36	70
QUESTÃO 14	18	QUESTÃO 37	74
QUESTÃO 15	20	QUESTÃO 38	77
QUESTÃO 16	22	QUESTÃO 39	80
QUESTÃO 17	24	QUESTÃO 40	84
QUESTÃO 18	26	QUESTÃO 41	87
QUESTÃO 19	29	QUESTÃO 42	90
QUESTÃO 20	31	QUESTÃO 43	91
QUESTÃO 21	33	QUESTÃO 44 (ANULADA)	93
QUESTÃO 22	37	QUESTÃO 45	95
QUESTÃO 23	40	QUESTÃO 46	98
QUESTÃO 24	43	QUESTÃO 48	102
QUESTÃO 25	46	QUESTÃO 49	105
QUESTÃO 26	48	QUESTÃO 50	107
QUESTÃO 27	50	QUESTÃO DISCURSIVA 3	109
QUESTÃO 28	53	QUESTÃO DISCURSIVA 4	112
QUESTÃO 29	55	QUESTÃO DISCURSIVA 5	115
QUESTÃO 30	57		

APRESENTAÇÃO

A principal razão para a existência de instituições é o provimento de serviços relevantes, necessários para o funcionamento da sociedade. O aprimoramento institucional é indispensável para a melhoria desses serviços e impacta positivamente levando a uma sociedade mais apta a enfrentar desafios conjuntos e assim a desenvolver-se.

Enquanto a melhoria da qualidade institucional é importante para todos os setores, entendemos a educação como um setor especial em sua capacidade de gerar impacto no desenvolvimento da sociedade. Assim, no caso da educação, entendemos que processos avaliativos, ao lado de importante investimento do Estado, revestem-se de caráter estratégico e são fundamentais para o país.

A avaliação do ensino de Graduação permite às instituições identificarem itens a melhorar, assim como contextualiza a capacidade da educação brasileira nesse nível perante outros países. No processo de avaliação proposto pelo Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira (INEP), o Exame Nacional de Desempenho de Estudantes (ENADE) tem papel central. O ENADE é fundamental como ferramenta avaliativa, pois ele mensura junto ao público discente o resultado do processo de ensino/aprendizagem. Ou seja, ele mede o resultado efetivo (percebido no aluno) de todo aparato de educação – desde a infraestrutura física institucional até o apoio ao docente, passando por metodologias e formas de interação com o público discente. Visto dessa perspectiva, entende-se que o ENADE se sobrepõe aos demais itens constituintes do processo avaliativo.

Os processos avaliativos são naturalmente envolventes. Isso se deve, por um lado, à síntese que ocorre traduzindo a qualidade de uma instituição através de um dado objetivo comparável e, por outro lado, à inerente concorrência entre instituições na sociedade moderna que inexoravelmente passam a utilizar dados sintéticos de avaliação. Assim, o resultado do ENADE, além dos efeitos benéficos já colocados acima, passa a ser utilizado mais comumente pela sociedade como indicativo para comparação de qualidade. Nesse momento o resultado da avaliação deixa de ser um dado para reflexão e melhoria institucional e passa a ser um elemento indutor na sociedade tanto da procura pelas instituições mais bem avaliadas quanto do entendimento da qualidade dos profissionais egressos formados pela instituição.

Deixando os desdobramentos desse aspecto para reflexão, o exercício do ENADE é extremamente positivo, pois se trata de mais um momento de oxigenação do corpo tanto discente quanto docente de uma instituição. Esses passam a entrar em contato com

problemas e formulações originadas por outros acadêmicos que estudam os mesmos temas na área de dedicação. Nesse contexto, o *ENADE Comentado – Computação* é um marco indispensável de posicionamento da Faculdade, por meio do conjunto de professores colaboradores, dentro do processo avaliativo, pois lança um olhar próprio, da Faculdade, ao exame da área. Esperamos que, além das saborosas discussões técnicas possíveis a partir deste volume, o mesmo sirva tanto a docentes como a discentes para o fortalecimento de sua identificação com a Faculdade de Informática através de seu curso de graduação, que, por fim, trata-se do elemento comum unindo a todos cuja avaliação é publicada à sociedade.

O *ENADE Comentado – Computação* foi construído com a colaboração dos professores referenciados ao longo do texto e com o cuidado especial dos organizadores do volume, professores Marcelo Hideki Yamaguti e Ana Paula Terra Bacelo. Agradecemos profundamente a todos pela contribuição à comunidade da Faculdade e, de forma especial, aos organizadores do volume.

Só nos resta desejar a todos boa leitura, boas discussões e reflexões técnicas, assim como boas discussões e reflexões acerca de nosso papel na sociedade, da importância da nossa competência conjunta e do ENADE nesse processo.

Bernardo Copstein
Coordenador Acadêmico
da FACIN

Fernando Luís Dotti
Diretor da FACIN

QUESTÃO 09

Seja A um conjunto e seja \sim uma relação entre pares de elementos de A .

Diz-se que \sim é uma relação de equivalência entre pares de elementos de A se as seguintes propriedades são verificadas, para quaisquer elementos a , a' e a'' de A :

- (i) $a \sim a$;
- (ii) se $a \sim a'$, então $a' \sim a$;
- (iii) se $a \sim a'$ e $a' \sim a''$, então $a \sim a''$.

Uma classe de equivalência do elemento a de A com respeito à relação \sim é o conjunto $\bar{a} = \{x \in A : x \sim a\}$.

O conjunto quociente de A pela relação de equivalência \sim é o conjunto de todas as classes de equivalência relativamente à relação \sim , definido e denotado como a seguir:

$$A/\sim = \{\bar{a} : a \in A\}.$$

A função $\pi: A \rightarrow A/\sim$ é chamada projeção canônica e é definida como $\pi(a) = \bar{a}, \forall a \in A$. Considerando as definições acima, analise as afirmações a seguir.

- I. A relação de equivalência \sim no conjunto A particiona o conjunto A em subconjuntos disjuntos: as classes de equivalência.
- II. A união das classes de equivalência da relação de equivalência \sim no conjunto A resulta no conjunto das partes de A .
- III. As três relações seguintes

$$\begin{matrix} \equiv \\ \equiv \\ \geq \end{matrix} (\text{mod } n)$$

são relações de equivalência no conjunto dos números inteiros \mathbb{Z} .

- IV. Qualquer relação de equivalência no conjunto A é proveniente de sua projeção canônica.

É correto apenas o que se afirma em

- A. II.
- B. III.

- C. I e III.
- D. I e IV.
- E. II e IV.

Resposta: alternativa (D)

Autores: Rafael Heitor Bordini e Alfio Ricardo de Brito Martini

COMENTÁRIO

No início da questão é dada a definição de uma **relação de equivalência**. Uma relação entre elementos de um mesmo conjunto é dita de equivalência se ela for **reflexiva**, como formalizado no item **I**, **simétrica**, como formalizado no item **II**, e **transitiva**, como formalizado no item **III**. A relação de igualdade entre expressões aritméticas, com a qual somos todos muito familiarizados, é um exemplo de uma relação de equivalência, em que vale, por exemplo: $5 = 5$ (por reflexividade); como $5 = 4 + 1$, então, $4 + 1 = 5$ (por simetria); $3 + 3 = 6$ e $6 = 2 + 4$, então, $3 + 3 = 2 + 4$ (por transitividade).

Toda relação de equivalência particiona o conjunto sobre a qual ela é definida em **classes de equivalência**. Por exemplo, a classe de equivalência do elemento 5, na notação dada $\overline{5}$, é o conjunto $\{5, 1+4, 6-1, (2*2)+1, 2+3, \dots\}$. Note que $1+4$ é o mesmo conjunto de 5. Note ainda que a definição de A/\sim afirma que, por exemplo, 5 e $1+4$ são elementos de A/\sim , mas como A/\sim é um conjunto, não há elementos repetidos. Portanto, A/\sim tem todas (e somente) as classes de equivalências diferentes em que a relação de igualdade particiona o conjunto A.

Pela assinatura da função π podemos ver que para um elemento de A ela retorna uma das classes de equivalência (que são todas elementos de A/\sim); a projeção retorna em particular aquela classe de equivalência da qual o elemento de A recebido como entrada faz parte. Ou seja, essa função π retorna para cada elemento de A a sua classe de equivalência em A, portanto no nosso exemplo $\pi(5) = \pi(1+4) = \{5, 1+4, 6-1, (2*2)+1, 2+3, \dots\}$, i.e., o conjunto que definimos acima.

Analisando, então, cada uma das afirmações da questão, temos:

- I. Toda relação de equivalência particiona o conjunto em classes de equivalência disjuntas, isso é sabido. No exemplo, veja que 5 e $4 + 1$ somente podem pertencer à mesma classe de equivalência junto a toda expressão cujo valor é 5 e não podem pertencer à classe daquelas que velem 6 ou 4 etc. A afirmativa está, portanto, correta.
- II. No nosso exemplo podemos imaginar que existem muitas classes de equivalências mas não é difícil perceber que há muito mais conjuntos no conjunto das partes do conjunto das expressões aritméticas. Por exemplo, $\{5\}$, $\{5, 4+1\}$, $\{4+1\}$, $\{5, 4+1, 2+3\}$, etc. são todos elementos do conjunto das partes, porém a união deles todos forma um único elemento do conjunto das classes de equivalência. Essa afirmativa, portanto, está incorreta.
- III. Como vimos no exemplo, igualdade é uma relação de equivalência. Da mesma forma que igualdade é uma relação de equivalência típica, desigualdades como \geq não são relações de equivalência. Veja, por exemplo, que $6 \geq 5$, mas não é o caso que $5 \geq 6$, portanto a relação não é reflexiva.

Assim já sabemos que a afirmativa é incorreta. Por curiosidade, vejamos que para $n=5$, $2 \equiv -8$ pois $2 - (-8) = 10$ e 5 é divisor de 10, da mesma forma $-8 \equiv 2$ pois 5 também divide -10 (i.e., $-8 - 2$). A relação é claramente reflexiva e transitiva também e, portanto, é relação de equivalência.

- IV. Para quaisquer $a, b \in A$, $a \sim b$ se e somente se $\pi(a) = \pi(b)$. Isto é, a relação de equivalência é recuperada tendo-se o mapeamento natural $\pi: A \rightarrow A/\sim$. A afirmativa está, portanto, correta.

REFERÊNCIAS

HEIN, JAMES. *Discrete Structures, Logic and Computability*. Jones and Bartlett, 2009.

PRATHER, Ronald. *Discrete Mathematical Structures for Computer Science*. Houghton Mifflin Harcourt, 1989.

ROSEN, Kenneth. *Discrete Mathematics and its Applications*. McGraw-Hill, 2011.

WECHLER, WOLFGANG. *Universal Algebra for Computer Scientists*. Nova York: Springer Verlag, 2012.

QUESTÃO 10

Em determinado período letivo, cada estudante de um curso universitário tem aulas com um de três professores, esses identificados pelas letras X, Y e Z. As quantidades de estudantes (homens e mulheres) que têm aulas com cada professor é apresentada na tabela de contingência abaixo.

	Professor X	Professor Y	Professor Z
Estudantes homens	45	5	32
Estudantes mulheres	67	2	4

A partir do grupo de estudantes desse curso universitário, escolhe-se um estudante ao acaso. Qual é a probabilidade de que esse estudante seja mulher, dado que ele tem aulas apenas com o professor X?

- A. $\frac{61}{73}$
- B. $\frac{61}{155}$
- C. $\frac{67}{155}$
- D. $\frac{22}{112}$
- E. $\frac{67}{112}$

Resposta: alternativa (E)

Autor: Hélio Radke Bittencourt

COMENTÁRIO

Esta é uma questão que pode ser resolvida de **maneira intuitiva**, sem o conhecimento formal da Teoria de Probabilidade. Quando a pergunta menciona “apenas ao professor X”, isso significa que apenas os alunos desse professor devem ser considerados ($45 + 67 = 112$). Portanto, dentre os 112 alunos do professor X, temos 67 mulheres, levando a escolha da alternativa **E**, $67/112$.

Formalmente, a solução da questão passaria pelo conhecimento de probabilidade condicional. Isso complicaria uma questão que é, por natureza, fácil.

Pela definição formal, teríamos:

$$P(Mulher | X) = \frac{P(Mulher \cap X)}{P(X)} = \frac{67/155}{112/155} = \frac{67}{155} \times \frac{155}{112} = \frac{67}{155} \times \frac{155}{112} = \frac{67}{112}$$

REFERÊNCIAS

BUSSAB, W. O; MORETTIN, P. A. *Estatística Básica*. 5. ed. São Paulo: Saraiva, 2002.

MEYER, Paul L. *Probabilidade: aplicações à Estatística*. Rio de Janeiro: LTC, 2000.

QUESTÃO 11

O problema da parada para máquinas de *Turing*, ou simplesmente problema da parada, pode ser assim descrito: determinar, para quaisquer máquinas de *Turing* M e palavra w , se M irá eventualmente parar com entrada w .

Mais informalmente, o mesmo problema também pode ser assim descrito: dados um algoritmo e uma entrada finita, decidir se o algoritmo termina ou se executará indefinidamente.

Para o problema da parada,

- A. existe algoritmo exato de tempo de execução polinomial para solucioná-lo.
- B. existe algoritmo exato de tempo de execução exponencial para solucioná-lo.
- C. não existe algoritmo que o solucione, não importa quanto tempo seja disponibilizado.
- D. não existe algoritmo exato, mas existe algoritmo de aproximação de tempo de execução polinomial que o soluciona, fornecendo respostas aproximadas.
- E. não existe algoritmo exato, mas existe algoritmo de aproximação de tempo de execução exponencial que o soluciona, fornecendo respostas aproximadas.

Resposta: alternativa (C)

Autora: Beatriz Regina Tavares Franciosi

COMENTÁRIO

A resposta correta é a que consta na alternativa **C**.

O problema da parada pode ser definido como:

Seja S o conjunto de todos os pares (A, D) , em que A é um algoritmo, e D , dado de entrada;

(A, D) tem a propriedade P se o algoritmo A , quando recebe o dado D , eventualmente produz um resultado (ou seja, eventualmente para).

A tese de Church-Turing mostra que o problema da parada é não decidível, ou seja, não existe um algoritmo H tal que para todo (A, D) que pertence à S :

$$H(A, D) = \begin{cases} 1 & \text{se } A(D) \text{ eventualmente para;} \\ 0 & \text{caso contrário} \end{cases}$$

A prova informal de que tal H não existe é obtida por contradição.

Suponha que H existe. Seja C o algoritmo: “entrada A ; executa $H(A, A)$; se $H(A, A)=0$, então, retorna 1, senão entra em loop”.

Então, $\forall A, (H(A, A)=0 \leftrightarrow \neg A(A) \text{ eventualmente para} \leftrightarrow H(A, A)=0)$ (pois H é função total) e $\forall A, (H(A, A)=0 \leftrightarrow \neg A(A) \text{ eventualmente para})$.

Tomando A como sendo C , obtemos que $C(C)$ eventualmente para, se e somente se $\neg C(C)$ eventualmente para, e isto é uma contradição!

Logo, não existe um algoritmo que solucione o problema.

As respostas das alternativas **A** e **B** não estão corretas, pois afirmam que existe um algoritmo que resolve o problema.

As respostas das alternativas **D** e **E** não estão corretas, pois afirmam que existe um algoritmo de aproximação e, pelo exposto na justificativa da resposta correta, tal algoritmo não existe.

REFERÊNCIAS

HOPCROFT, John E.; MOTWANI, Rajeev; ULLMAN, Jeffrey D. *Introdução à teoria de autômatos, linguagens e computação*. Elsevier, 2003.

BOOLOS, George S.; JEFFREY, Richard C. *Computability and logic*. 3. ed. Cambridge University Press, 1990.

QUESTÃO 12

Considere a gramática a seguir, em que S , A e B são símbolos não terminais, 0 e 1 são terminais e ε é a cadeia vazia.

$$S \rightarrow 1S|0A|\varepsilon$$

$$A \rightarrow 1S|0B|\varepsilon$$

$$B \rightarrow 1S|\varepsilon$$

A respeito dessa gramática, analise as afirmações a seguir.

- I. Nas cadeias geradas por essa gramática, o último símbolo é 1 .
- II. O número de zeros consecutivos nas cadeias geradas pela gramática é, no máximo, dois.
- III. O número de uns em cada cadeia gerada pela gramática é maior que o número de zeros.
- IV. Nas cadeias geradas por essa gramática, todos os uns estão à esquerda de todos os zeros.

É correto apenas o que se afirma em

- A. I.
- B. II.
- C. I e III.
- D. II e IV.
- E. III e IV.

Resposta: alternativa (B)

Autor: Júlio Henrique Araújo Pereira Machado

COMENTÁRIO

A questão apresentada está relacionada aos conceitos de Gramáticas Regulares e Linguagens Regulares da disciplina de Linguagens Formais. É uma questão de fácil resolução a partir da correta aplicação das regras de derivação da gramática para a construção de palavras.

A técnica a ser utilizada para a justificativa de uma alternativa como falsa é a apresentação de um contraexemplo de sequência de derivações que demonstre a falsidade.

A afirmativa I deve ser considerada falsa. O contraexemplo é a geração da palavra-vazia a partir do símbolo inicial S (nota do autor: essa é uma suposição, já que a questão pode ser considerada mal construída já que não informa qual dos símbolos, S, A ou B, é o símbolo inicial).

$S \Rightarrow \varepsilon$ (aplicação da regra $S \rightarrow \varepsilon$)

A afirmativa II é verdadeira. Considere a seguinte sequência de derivações, na qual W representa uma cadeia de símbolos terminais. A derivação demonstra as duas únicas regras que podem ser aplicadas a qualquer momento para remover o símbolo terminal B da palavra sendo gerada, de forma que número de zeros consecutivos será sempre no máximo dois.

$S \Rightarrow^* W0A$

$\Rightarrow W00B$ (Aplicação da regra $S \rightarrow 0B$ é a única que gera zeros seguidos.)

$\Rightarrow W001S$ (Aplicação da regra $B \rightarrow 1S$.)

ou

$\Rightarrow W00$ (Aplicação da regra $B \rightarrow \varepsilon$.)

A afirmativa III é trivialmente falsa, já que a palavra-vazia pertence ao conjunto de palavras da linguagem gerada pela gramática apresentada. Ou seja, a quantidade zero de símbolos uns e zeros torna a afirmativa falsa. A seguinte sequência de derivações é o contraexemplo que justifica a afirmativa.

$S \Rightarrow \varepsilon$ (aplicação da regra $S \rightarrow \varepsilon$)

A afirmativa IV deve ser considerada falsa, pois é possível gerar a palavra 01 (na qual uns aparecem à direita de zeros) a partir da seguinte sequência de derivações do contraexemplo.

$S \Rightarrow 0A$ (Aplicação da regra $S \rightarrow 0A$.)

$S \Rightarrow 01S$ (Aplicação da regra $A \rightarrow 1S$.)

$S \Rightarrow 01$ (Aplicação da regra $S \rightarrow \varepsilon$.)

REFERÊNCIA

HOPCROFT, John E; ULLMAN, Jeffrey D; MOTWANI, Rajeev. *Introdução à teoria de autômatos, linguagens e computação*. Rio de Janeiro: Elsevier, 2003.

QUESTÃO 13 (ANULADA)

O problema do escalonamento de intervalos tem como entrada um conjunto de intervalos numéricos (usualmente interpretados como início e fim de atividades), e o objetivo é escolher, desse conjunto, o maior número possível de intervalos disjuntos dois a dois. Há vários problemas práticos que podem ser modelados dessa forma, como, por exemplo, a seleção de tarefas com horário marcado.

O problema do escalonamento de intervalos pode ser resolvido com o algoritmo descrito a seguir. O conjunto de intervalos dados inicialmente é R e o conjunto de intervalos escolhidos, A , começa vazio.

```
enquanto  $R$  não estiver vazio,  
    seja  $x$  o intervalo de  $R$  com menor tempo de término, e que não tenha  
    interseção com algum intervalo em  $A$   
    retire  $x$  de  $R$  e adicione ao conjunto  $A$  retorne  $A$ 
```

A respeito desse algoritmo, analise as seguintes asserções. Para checar se o algoritmo está correto, basta verificar que o primeiro intervalo adicionado ao conjunto A necessariamente faz parte de uma solução ótima.

PORQUE

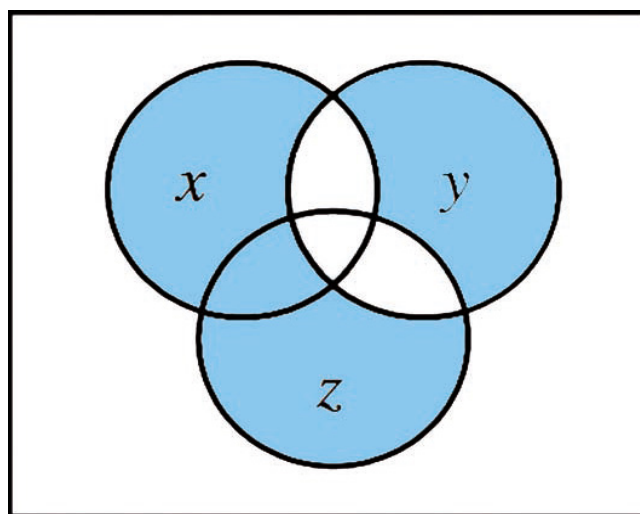
Pode-se mostrar, por indução no número máximo de intervalos calculados (ou seja, no número de vezes que o laço “enquanto” é executado), que, embora possa haver soluções tão boas quanto A , nenhuma delas é estritamente melhor que A . O conjunto com um único intervalo é a base de indução.

Acerca dessas asserções, assinale a opção correta.

- A. As duas asserções são proposições verdadeiras, e a segunda é uma justificativa correta da primeira.
- B. As duas asserções são proposições verdadeiras, mas a segunda não é uma justificativa correta da primeira.
- C. A primeira asserção é uma proposição verdadeira, e a segunda, uma proposição falsa.
- D. A segunda asserção é uma proposição falsa e a segunda, uma proposição verdadeira.
- E. Tanto a primeira quanto a segunda asserções são proposições falsas.

QUESTÃO 14

Observe o diagrama de Venn a seguir.



A função representada em azul no diagrama também poderia ser expressa pela função lógica $f(x, y, z) =$

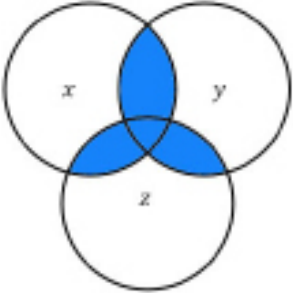
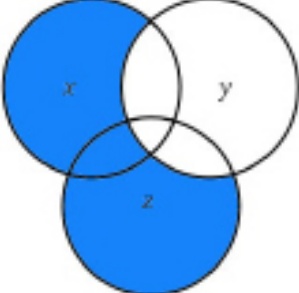
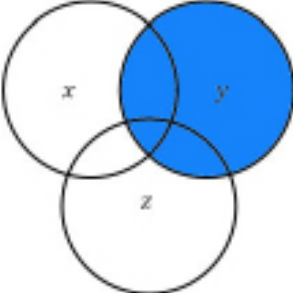
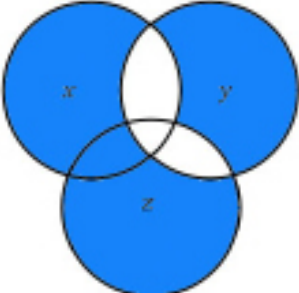
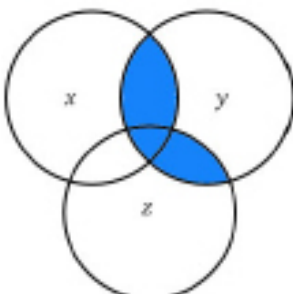
- A. $(x + z) y + x \bar{y} z$
- B. $(x + z) y + \bar{x} y \bar{z}$
- C. $(x + z) y + \bar{x} \bar{y} \bar{z}$
- D. $(x + z) \bar{y} + x \bar{y} z$
- E. $(x + z) \bar{y} + \bar{x} y \bar{z}$

Resposta: alternativa (E)

Autor: Filipe Jaeger Zabala

COMENTÁRIO

Esta questão aplica conceitos de Teoria de Conjuntos e Funções (IEZZI; MURAKAMI, 2013, 77-A e 78-A). Inicialmente aplica-se a propriedade distributiva no primeiro termo da função em cada item, na forma geral , equivalente à distributiva da intersecção em relação à união , (IEZZI; MURAKAMI, 2013, 31-A). Pode-se então descrever as relações de maneira mais intuitiva através diagramas de Venn e da notação de conjunto:

<p>a) $(x \cap y) \cup (y \cap z) \cup (x \cap \bar{y} \cap z)$</p> 	<p>d) $(x \cap \bar{y}) \cup (\bar{y} \cap z) \cup (x \cap \bar{y} \cap z)$</p> 
<p>b) $(x \cap y) \cup (y \cap z) \cup (\bar{x} \cap y \cap z)$</p> 	<p>e) $(x \cap \bar{y}) \cup (\bar{y} \cap z) \cup (\bar{x} \cap y \cap z)$</p> 
<p>c) $(x \cap \bar{y}) \cup (\bar{y} \cap z) \cup (\bar{x} \cap y \cap \bar{z})$</p> 	<p>Assim, encontra-se</p> $\begin{aligned} &(x \cap \bar{y}) \cup (\bar{y} \cap z) \cup (\bar{x} \cap y \cap \bar{z}) \\ &\quad \Downarrow \\ &(x \cup z) \cap \bar{y} \cup (\bar{x} \cap y \cap \bar{z}) \\ &\quad \Downarrow \\ &f(x, y, z) = (x + z)\bar{y} + \bar{x}y\bar{z}. \end{aligned}$ <p>Alternativa E.</p>

REFERÊNCIAS

ALENCAR FILHO, E. *Teoria elementar dos conjuntos*. São Paulo: Nobel, 1976.

IEZZI, G; MURAKAMI, C. *Matemática Elementar – Conjuntos e Funções*. 9. ed. São Paulo: Editora Atual, 2013.

QUESTÃO 15

Suponha que seja necessário desenvolver uma ferramenta que apresente o endereço IP dos múltiplos roteadores, salto a salto, que compõem o caminho do hospedeiro em que a ferramenta é executada até um determinado destino (segundo seu endereço IP), assim como o *round-trip time* até cada roteador. Tal ferramenta precisa funcionar na *Internet* atual, sem demandar mudanças em roteadores nem a introdução de novos protocolos.

Considerando o problema acima, qual dos seguintes protocolos representaria a melhor (mais simples e eficiente) solução?

- A. IP: Internet Protocol.
- B. UDP: User Datagram Protocol.
- C. TCP: Transmission Control Protocol.
- D. ICMP: Internet Control Message Protocol.
- E. DHCP: Dynamic Host Configuration Protocol.

Resposta: alternativa (D)

Autoras: Ana Cristina Benso da Silva e Cristina Moreira Nunes

COMENTÁRIO

A questão solicita que se identifique qual dos protocolos, vigentes na pilha TCP/IP, pode ser utilizado na construção de uma ferramenta que trace o caminho entre o hospedeiro (origem) e um destino, identificando o endereço IP de cada roteador intermediário e o *round-trip time* até o mesmo.

O gabarito apresenta a alternativa **D** como resposta, ou seja, o protocolo ICMP. O protocolo ICMP é o protocolo de controle da pilha TCP/IP, que gera mensagens de erro para informar a uma origem o motivo pelo qual os datagramas enviados não chegaram ao seu destinatário, bem como gera mensagens que permitem testar a alcançabilidade de um destino.

Assim, o hospedeiro pode enviar mensagens do tipo *ICMP Echo Request* para um destinatário, variando o TTL (*Time to Live*) do datagrama IP, ou seja, a primeira mensagem terá TTL igual 1, a segunda, TTL igual 2, e assim sucessivamente até alcançar o destino. Essas mensagens serão processadas a cada roteador intermediário (salto a salto). Em cada roteador o TTL será decrementado de 1 (um) salto

e, naquele em que o decremento resultar em 0 (zero), o datagrama será descartado e será enviada uma mensagem *ICMP Time to Live Exceeded in Transit* para o hospedeiro. Nesta mensagem, o IP de origem, no datagrama IP, é o endereço do roteador que realizou o descarte. O *round-trip time* tem de ser controlado pela aplicação, ou seja, a aplicação deve manter um *timestamp* das mensagens enviadas e o *timestamp* para as mensagens recebidas, a fim de calcular o tempo.

Após passar pelo último roteador, com TTL igual ao número de roteadores mais 1 (um), a mensagem chegará ao *host* destino. Nesse caso, o *host* não fará o decremento do TTL e o descarte do datagrama. O *host* irá responder à mensagem gerada pela origem, que continha um *ICMP Echo Request*, enviando uma mensagem do tipo *ICMP Echo Reply*. A origem poderá então constatar que atingiu o destino.

REFERÊNCIAS

COMER, Douglas. *Internetworking with TCP/IP*. 6. ed. Boston: Addison-Wesley, 2013, v. 1. 744p.

TANENBAUM, A. *Computer Networks*. 5. ed. Upper Saddle River: Prentice Hall, 2010. 960p.

QUESTÃO 16

Um navegador *Web* executa em um hospedeiro **A**, em uma rede de uma organização, e acessa uma página localizada de um servidor *Web* em um hospedeiro **B**, situado em outra rede na *Internet*. A rede em que **A** se situa conta com um servidor DNS local. Um profissional deseja fazer uma lista com a sequência de protocolos empregados e comparar com o resultado apresentado por uma ferramenta de monitoramento executada no hospedeiro **A**. A lista assume que

- I. todas as tabelas com informações temporárias e caches estão vazias;
- II. o hospedeiro cliente está configurado com o endereço IP do servidor DNS local.

Qual das sequências a seguir representa a ordem em que mensagens, segmentos e pacotes serão observados em um meio físico ao serem enviados pelo hospedeiro **A**?

- A.** ARP, DNS/UDP/IP, TCP/IP e HTTP/TCP/IP.
- B.** ARP, DNS/UDP/IP, HTTP/TCP/IP e TCP/IP.
- C.** DNS/UDP/IP, ARP, HTTP/TCP/IP e TCP/IP.
- D.** DNS/UDP/IP, ARP, TCP/IP e HTTP/TCP/IP.
- E.** HTTP/TCP/IP, TCP/IP, DNS/UDP/IP e ARP.

Resposta: alternativa (A)

Autoras: Cristina Moreira Nunes e Ana Cristina Benso da Silva

COMENTÁRIO

Quando tentamos acessar uma página Web, o protocolo HTTP é utilizado para encapsular a requisição. Esse protocolo utiliza o protocolo TCP como transporte, que é encapsulado no protocolo IP e posteriormente no protocolo utilizado pela placa de rede. Para a requisição HTTP poder ser enviada pela máquina, uma série de passos anteriores são necessários:

1º) o nome do servidor Web digitado no browser pelo usuário precisa ser mapeado em seu endereço IP através do DNS. Como a consulta ao servidor de DNS precisa passar pela rede, o protocolo ARP é utilizado para descobrir o endereço MAC da próxima máquina que deve receber a requisição. Assim, este protocolo (ARP) será o primeiro a ser transmitido pela máquina do usuário;

2º) sabendo o endereço MAC da próxima máquina, é possível utilizar o protocolo DNS para fazer o mapeamento do nome do servidor Web em seu endereço IP. Para a mensagem do protocolo DNS sair da máquina do usuário, a mesma é encapsulada no protocolo UDP, o qual é encapsulado pelo protocolo IP;

3º) depois de descoberto o endereço IP do servidor Web, a requisição do protocolo HTTP pode ser montada. Contudo, como o protocolo HTTP utiliza o protocolo TCP como transporte e o protocolo TCP é orientado à conexão, o estabelecimento da conexão TCP com o servidor Web precisa ser realizado. As mensagens enviadas pelo protocolo TCP são sempre encapsuladas no protocolo IP;

4º) ao ser estabelecida a conexão TCP com o servidor Web, a requisição do protocolo HTTP pode ser enviada. Esta requisição é encapsulada pelo protocolo TCP, que é encapsulado pelo protocolo IP.

REFERÊNCIAS

COMER, D. E. *Interligação de redes com TCP/IP*. Rio de Janeiro: Campus, 2006, v. 1.

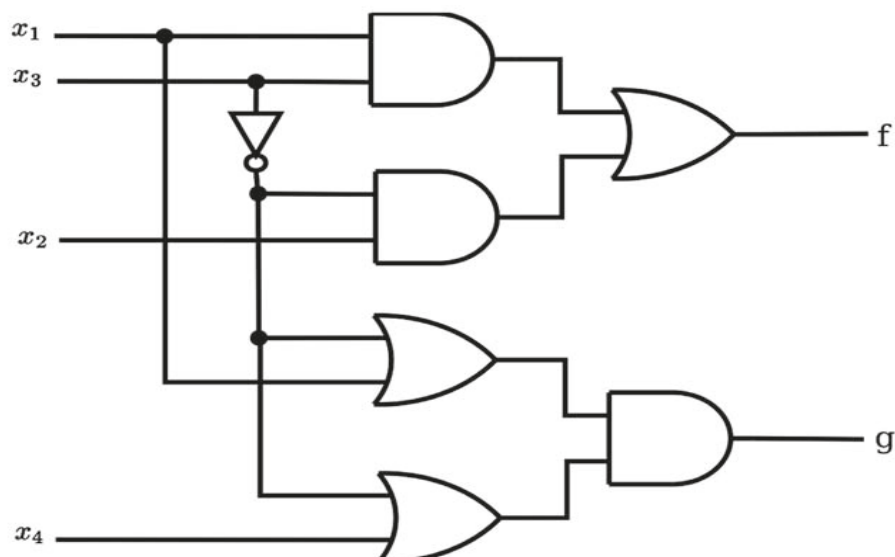
TANENBAUM, A.; WETHERALL, D. *Redes de Computadores*. 5. ed. Pearson Education, 2011.

QUESTÃO 17

A tabela a seguir apresenta a relação de mintermos e maxtermos para três variáveis.

Linha	x_1	x_2	x_3	Mintermo	Maxtermo
0	0	0	0	$m_0 = \bar{x}_1 \bar{x}_2 \bar{x}_3$	$M_0 = x_1 + x_2 + x_3$
1	0	0	1	$m_1 = \bar{x}_1 \bar{x}_2 x_3$	$M_1 = x_1 + x_2 + \bar{x}_3$
2	0	1	0	$m_2 = \bar{x}_1 x_2 \bar{x}_3$	$M_2 = x_1 + \bar{x}_2 + x_3$
3	0	1	1	$m_3 = \bar{x}_1 x_2 x_3$	$M_3 = x_1 + \bar{x}_2 + \bar{x}_3$
4	1	0	0	$m_4 = x_1 \bar{x}_2 \bar{x}_3$	$M_4 = \bar{x}_1 + x_2 + x_3$
5	1	0	1	$m_5 = x_1 \bar{x}_2 x_3$	$M_5 = \bar{x}_1 + x_2 + \bar{x}_3$
6	1	1	0	$m_6 = x_1 x_2 \bar{x}_3$	$M_6 = \bar{x}_1 + \bar{x}_2 + x_3$
7	1	1	1	$m_7 = x_1 x_2 x_3$	$M_7 = \bar{x}_1 + \bar{x}_2 + \bar{x}_3$

Analise o circuito de quatro variáveis a seguir.



Considerando esse circuito, as funções **f** e **g** são, respectivamente,

- A. $\Sigma m(0,1,2,3,6,7,8,9)$ e $\Sigma m(2,3,6,7,10,14)$.
- B. $\Sigma m(4,5,10,11,12,13,14,15)$ e $\Sigma m(0,1,4,5,8,9,11,12,13,15)$.

- C. $\Pi(0,1,2,3,6,7,8,9)$ e $\Pi(0,1,4,5,8,9,11,12,13,15)$.
- D. $\Pi(4,5,10,11,12,13,14,15)$ e $\Sigma(2,3,6,7,10,14)$.
- E. $\Pi(4,5,10,11,12,13,14,15)$ e $\Pi(2,3,6,7,10,14)$.

Resposta: alternativa (B)

Autor: César Augusto Missio Marcon

COMENTÁRIO

Ao propagar as entradas pelas portas lógicas do circuito apresentado, a função **f** pode ser capturada por uma soma de produtos, enquanto a função **g**, por um produto de somas:

$$f = x_1 x_3 + x_2 \overline{x_3}$$

$$g = (x_1 + \overline{x_3}) \cdot (\overline{x_3} + x_4)$$

Um mintermo de uma função pode ser expandido com todos os componentes que ele representa, o que equivale a dizer todas as combinações implicitamente representadas no mintermo. Por exemplo, para um universo de duas variáveis A e B, se uma função q é igual a apenas A ($q = A$), então, todas as combinações de B estão implícitas na equação, o que equivale a dizer que $q = A\overline{B} + AB$.

Reescrevendo os mintermos de **f** temos:

$$x_1 x_3 \rightarrow x_1 \overline{x_2} \overline{x_3} \overline{x_4} + x_1 \overline{x_2} \overline{x_3} x_4 + x_1 x_2 \overline{x_3} \overline{x_4} + x_1 x_2 \overline{x_3} x_4 \rightarrow \Sigma m(10, 11, 14, 15)$$

$$x_2 \overline{x_3} \rightarrow \overline{x_1} x_2 \overline{x_3} \overline{x_4} + \overline{x_1} x_2 \overline{x_3} x_4 + x_1 x_2 \overline{x_3} \overline{x_4} + x_1 x_2 \overline{x_3} x_4 \rightarrow \Sigma m(4, 5, 12, 13)$$

Logo:

$$f = \Sigma m(4, 5, 10, 11, 12, 13, 14, 15)$$

Multiplicando os produtos de **g**, este passa a ser representado por uma soma de produtos:

$$g = x_1 \overline{x_3} + \overline{x_3} + \overline{x_3} x_4 + x_1 x_4$$

Tendo em vista que o mintermo com apenas $\overline{x_3}$ contém todos os demais mintermos que têm esta mesma entrada, então, **g** pode ser reescrita como:

$$g = \overline{x_3} + x_1 x_4$$

Reescrevendo os mintermos de **g**, temos:

$$\overline{x_3} \rightarrow \overline{x_1} \overline{x_2} \overline{x_3} \overline{x_4} + \overline{x_1} \overline{x_2} \overline{x_3} x_4 + \overline{x_1} x_2 \overline{x_3} \overline{x_4} + \overline{x_1} x_2 \overline{x_3} x_4$$

$$x_1 \overline{x_2} \overline{x_3} \overline{x_4} + x_1 \overline{x_2} \overline{x_3} x_4 + x_1 x_2 \overline{x_3} \overline{x_4} + x_1 x_2 \overline{x_3} x_4 \rightarrow \Sigma m(0,1,4,5,8,9,12,13)$$

$$x_1 x_4 \rightarrow x_1 \overline{x_2} \overline{x_3} x_4 + x_1 \overline{x_2} x_3 x_4 + x_1 x_2 \overline{x_3} x_4 + x_1 x_2 x_3 x_4 \rightarrow \Sigma m(9, 11, 13, 15)$$

Logo:

$$g = \Sigma m(0, 1, 4, 5, 8, 9, 11, 12, 13, 15)$$

REFERÊNCIAS

PEDRONI, Voleni A. *Eletrônica Digital Moderna e VHDL*. Rio de Janeiro: Elsevier, 2011. 620p.

QUESTÃO 18

Um vendedor de artigos de pesca obteve com um amigo o código executável (já compilado) de um programa que gerencia vendas e faz o controle de estoque, com o intuito de usá-lo em sua loja. Segundo o seu amigo, o referido programa foi compilado em seu sistema computacional pessoal (sistema A) e funciona corretamente. O vendedor constatou que o programa executável também funciona corretamente no sistema computacional de sua loja (sistema B). Considerando a situação relatada, analise as afirmações a seguir.

- I. Os computadores poderiam ter quantidades diferentes de núcleos (cores).
- II. As chamadas ao sistema (system call) do sistema operacional no sistema A devem ser compatíveis com as do sistema B.
- III. O conjunto de instruções do sistema A poderia ser diferente do conjunto de instruções do sistema B.
- IV. Se os registradores do sistema A forem de 64 bits, os registradores do sistema B poderiam ser de 32 bits.

É correto o que se afirma em

- A. III, apenas.
- B. I e II, apenas.
- C. III e IV, apenas.
- D. I, II e IV, apenas.
- E. I, II, III e IV.

Resposta: alternativa (B)

Autor: Edson Ifarraguirre Moreno

COMENTÁRIO

Um programa/arquivo/código executável é gerado para um sistema computacional específico. Aqui, entende-se um sistema computacional como o conjunto básico composto por um processador-alvo, ou família de processadores compatíveis (e.g., x86), e um sistema operacional-alvo, ou família de sistema operacional compatível (e.g., Linux).

Um programa executável é normalmente obtido a partir de uma descrição realizada a partir de uma linguagem de alto nível de abstração, tal como C++, chamada de código-fonte. Esse código é normalmente implementado de forma independente do sistema computacional alvo. O vínculo com o sistema computacional alvo se dá quando há uso de um compilador para obtenção do código executável. Um compilador é uma ferramenta responsável pela tradução do código fonte, de alto nível de abstração, em um código objeto, o qual está vinculado a um sistema operacional específico e processador específico. Tal vínculo é fundamental, pois o código executável gerado deve conter uma estrutura tal que o permita ser gerenciável pelo sistema operacional, bem como ser compreendido e executado pelo processador e pelo sistema operacional.

Sob o ponto de vista do formato do arquivo, cada sistema operacional possui uma forma de interpretar um código executável. Minimalistamente sendo dito, todo código executável deve possuir dois blocos principais: (a) um de informações acerca do próprio arquivo, tal como organização do arquivo e tamanho dos blocos internos; e (b) um com as instruções e dados que compõem o programa a ser executado. A estrutura inicial do arquivo deve ser interpretada pelo sistema operacional quando um programa executável é “lançado” (e.g., duplo clique em um arquivo .exe no Windows) a fim de que o programa possa ser gerenciado pelo sistema operacional. A segunda estrutura do arquivo deve ser tal que reflita exatamente o conjunto de instruções suportadas por um processador-alvo.

Cada processador é idealizado para dar suporte a um vocabulário finito, conhecido no universo de arquitetura de computadores como arquitetura de conjunto de instruções (em inglês, *Instruction Set Architecture*, ISA) ou simplesmente conjunto de instruções. É comum que processadores distintos tenham ISAs distintas e que estes sejam incompatíveis. Um exemplo facilmente encontrável na internet são as versões de programas executáveis para processadores AMD e para processadores x86. O trecho de código contido no programa executável nada mais é do que um conjunto de representações binárias as quais têm relação direta com as instruções suportadas por um determinado processador. Como exemplo, imagine que uma determinada instrução, contida do trecho de código do programa executável, contenha o valor binário 0x123456 e que este programa fora compilado para o processador xABC. Neste processador, esta instrução teria uma relação de uma operação de soma entre um dado par de registradores. Mas, se este mesmo código fosse executado em um processador xYZT, a mesma instrução 0x123456 teria relação com outra operação, tal como uma chamada de sistema. Dessa forma, a interpretação de todo o código no processador xYZT, que não fora planejado como o processador-alvo, deveria levar a uma execução que não faria sentido, executando operações que, postas em sequência, não gerariam uma computação válida.

Levando-se em consideração o que foi comentado e revendo cada uma das afirmações que foram apresentadas, pode-se afirmar que:

I – Os computadores poderiam ter a quantidades diferentes de núcleos (cores). Verdadeiro, mas desde que os elementos de processamento (*i.e.*, processadores/núcleos) fossem equivalentes quanto a sua ISA. O controle do número de processadores é de responsabilidade do sistema operacional. Certamente que o conhecimento de que um dado programa será executado em um sistema de múltiplos processadores permitiria o planejamento deste para o melhor aproveitamento de tal característica.

II – As chamadas de sistema (system call) do sistema operacional do sistema A devem ser compatíveis com o do sistema B. Verdadeiro, pois a incompatibilidade entre chamadas de sistema poderia fazer com que o programa não rodasse de forma equivalente em ambos os sistemas. Uma chamada de sistema é a forma como um programa, sendo executado por um processador, interage com o sistema operacional. Chamadas de sistema podem ser empregadas para diferentes fins, tais como a

sinalização da ocorrência de um erro em um programa (e.g., uma divisão por zero) ou a requisição de uma leitura de um arquivo de texto armazenado em um disco rígido. Todavia, a forma como tal interação ocorre, ou é iniciada, varia de processador para processador. Daí a necessidade de compatibilidade.

III – O conjunto de instruções do sistema A pode ser diferente do conjunto de instruções do sistema B. Falso. Como dito anteriormente, o conjunto de instruções (ISA) é a linguagem entendida pelo processador. Um programa somente pode ser executado corretamente pelo processador-alvo caso o conjunto de instruções mapeados nesse programa seja compatível. De forma análoga, imagine que um programa executável cujas instruções foram todas mapeadas para palavras em português. Dessa forma, todo processador que vá interpretar tal programa deve ter em seu vocabulário termos em português. Se o processador-alvo conhece tão somente vocabulário em chinês, a execução desse programa não ocorrerá conforme o esperado. Como no texto da Questão 18 o programa que foi executado é compatível entre os dois sistemas, a afirmação **III** não é verdadeira.

IV – Se os registradores do sistema A forem de 64 bits, os registradores do sistema B poderiam ser de 32. Falso. Arquiteturas de processadores que trabalham com registradores de 64 bits possuem instruções que dão suporte a tal tipo de característica, ou seja, possuem instruções que permitem a operação diretamente em 32 bits. Normalmente, por serem uma evolução de arquiteturas anteriores elaboradas sobre 32 bits, tais arquiteturas suportam operações com representações de 32 bits. O contrário não é verdadeiro. Sendo assim, o conjunto de instruções, e forma de operação, de processadores de 64 bits suporta programas compilados para essa arquitetura e, muitas vezes, para arquiteturas de processadores de mesma ISA no que se refere às operações em 32 bits. Já arquiteturas de 32 bits não dão suporte a arquiteturas de 64 bits.

Como conclusão, a alternativa **B** é a correta, ou seja, as afirmativas **I** e **II** estão corretas.

REFERÊNCIAS

DE ROSE, Cesar A. F.; NAVAUX, Philippe O. A. *Arquiteturas paralelas*. Porto Alegre: Sagra Luzzatto, 2003. 152 p. Série Livros Didáticos; 15.

SILBERSCHATZ, Abraham; GALVIN, Peter B.; GAGNE, Greg. *Operating system concepts*. 7. ed. John Wiley & sons, 2004. 887p.

STALLINGS, W. *Arquitetura e Organização de Computadores*. 8. ed. São Paulo: Pearson, 2010. 640 p.

QUESTÃO 19

Uma equipe está realizando testes com base nos códigos-fonte de um sistema. Os testes envolvem a verificação de diversos componentes individualmente, bem como das interfaces entre os componentes.

No contexto apresentado, essa equipe está realizando testes em nível de

- A. unidade.
- B. aceitação.
- C. sistema e aceitação.
- D. integração e sistema.
- E. unidade e integração.

Resposta: alternativa (E)

Autor: Flávio Moreira de Oliveira

COMENTÁRIO

Os níveis de teste são níveis de abstração/granularidade em que são realizadas as atividades de teste de software. A definição do nível em que um teste será realizado determina as fronteiras do artefato de software que será testado; em consequência, determina o foco do teste, os defeitos que serão investigados, o espaço de entrada que será considerado para seleção dos casos de teste, as técnicas de teste aplicadas e a descrição dos resultados esperados. Tradicionalmente, consideram-se três níveis:

- A. nível unitário ou de unidade;
- B. nível de integração;
- C. nível de sistema.

No nível unitário, o artefato a ser testado é uma unidade do software, ou seja, os componentes testáveis mais simples do sistema (para um componente ser testável, é preciso que implemente um comportamento ou funcionalidade identificável). A definição do que será considerado como “unidade” depende do paradigma de programação utilizado; no paradigma orientado a objetos, considera-se como unidade a classe (poder-se-ia argumentar que um método já implementaria um comportamento; no entanto, a experiência demonstra que organizar o teste por classes é mais produtivo, pois frequentemente

os métodos em uma mesma classe interagem entre si para realizar um serviço). Já no paradigma imperativo, a unidade pode ser um procedimento ou um módulo, dependendo da complexidade. Os defeitos investigados são de codificação e o espaço de entrada é de chamadas – ou sequências de chamadas – de métodos/funções do componente em teste. Podem ser usadas técnicas funcionais ou estruturais, envolvendo análise do código-fonte. Assim, podemos dizer que o teste no nível unitário é realizado na perspectiva do desenvolvedor, em um nível de abstração equivalente ao da atividade de codificação.

No nível de integração, o artefato a ser testado é um *subsistema*. Tipicamente, a integração é um processo incremental, integrando componentes em subsistemas e testando-os, depois integrando subsistemas entre si e testando-os novamente, até construir uma versão completa do sistema. O foco do teste é a investigação de defeitos nos relacionamentos entre os componentes e subsistemas, que são mais difíceis de detectar no teste unitário. Podem ser reutilizados casos de teste do nível unitário, adicionando outros para exercitar interações entre os componentes. Também aqui podem ser usadas técnicas funcionais e estruturais, porém o teste baseado em modelos tem papel importante no nível de integração. Isso porque a especificação dos relacionamentos entre componentes é uma informação de *projeto*, descrita nos diferentes modelos (lógicos, estruturais, comportamentais) do sistema. Pode-se dizer, portanto, que o teste no nível unitário é realizado na perspectiva do projetista ou arquiteto.

No nível de sistema, o artefato em teste é o sistema completo; podemos entender que as etapas incrementais do nível de integração culminam no nível de sistema, quando todos os subsistemas estão integrados. O foco do teste aqui é a investigação de defeitos do ponto de vista do usuário. Assim, o espaço de entrada é determinado pelos requisitos propriamente ditos do sistema, e as técnicas funcionais serão as mais utilizadas. Também no nível de sistema é usual testar os chamados requisitos não funcionais: desempenho, confiabilidade, usabilidade etc.

Pelo exposto anteriormente, fica evidente que a resposta correta à questão é a alternativa **E**, dado que os testes descritos “envolvem a verificação de diversos componentes individualmente, bem como das interfaces entre os componentes”. Logo, são testes no nível unitário e de integração.

REFERÊNCIAS

DELAMARO, M. E.; MALDONADO, J. C.; JINO, M. *Introdução ao Teste de Software*. 1. ed. Rio de Janeiro: Elsevier, 2007. 394p.

PEZZÉ, M; YOUNG M. *Teste e Análise de Software: Processo, Princípios e Técnicas*. Porto Alegre: Bookman, 2008. 512p.

QUESTÃO 20

Considere que G é um grafo qualquer e que V e E são os conjuntos de vértices e de arestas de G , respectivamente. Considere também que $\text{grau}(v)$ é o grau de um vértice v pertencente ao conjunto V . Nesse contexto, analise as seguintes asserções.

Em G , a quantidade de vértices com grau ímpar é ímpar.

PORQUE

Para G , vale a identidade dada pela expressão

$$\sum_{v \in V} \text{grau}(v) = 2 |E|$$

Acerca dessas asserções, assinale a opção correta.

- A. As duas asserções são proposições verdadeiras, e a segunda é uma justificativa correta da primeira.
- B. As duas asserções são proposições verdadeiras, mas a segunda não é uma justificativa correta da primeira.
- C. A primeira asserção é uma proposição verdadeira, e a segunda uma proposição falsa.
- D. A primeira asserção é uma proposição falsa, e a segunda uma proposição verdadeira.
- E. Tanto a primeira quanto a segunda asserções são proposições falsas.

Resposta: alternativa (D)

Autor: Marco Aurélio Souza Mangan

COMENTÁRIO

O contexto da questão é formado por definições matemáticas. **Primeiro a definição de grafo e em seguida o uso de termos como vértices, arestas, conjuntos e das relações grau, somatório e aridade.** Alguns autores reservam aresta para grafos não dirigidos, entretanto, consideramos o uso mais amplo, no qual as arestas podem ser dirigidas ou não. Nesse caso, temos que considerar tanto grafos dirigidos quanto não dirigidos ao avaliar as asserções. A definição de grau não é fornecida e é essencial para responder a questão.

Segundo Cormen *et al.* (2009), o grau de um vértice em um grafo não direcionado é o número de arestas incidentes. Ainda segundo os mesmos autores, o grau de um vértice em um grafo direcionado é a soma dos graus de entrada e de saída do vértice.

Imaginemos um grafo $G(V, E)$, sendo $V = \{a, b\}$ e $E = \{(a, b)\}$, os dois vértices têm grau ímpar. Nesse caso, temos um grafo que invalida a primeira asserção. Existe apenas uma aresta, então $\text{grau}(a) = 0 + 1 = 1$ e $\text{grau}(b) = 1 + 0 = 1$, no caso de grafos direcionados. No caso de grafos não direcionados, $\text{grau}(a) = 1$ e $\text{grau}(b) = 1$, o que mantém o resultado. A validade da asserção é determinada por um contraexemplo, como é o caso da maior das questões que solicita a avaliação de uma regra universal. Com isso, são eliminadas as alternativas A, B e C.

Toda aresta conecta dois vértices, logo, cada aresta contribui com o grau total do grafo duas vezes, uma em cada vértice que conecta. Portanto, a segunda asserção é verdadeira. Existe um caso particular, em que uma aresta pode conectar um vértice a ele mesmo, formando um arco. Neste caso a contribuição se acumula em um mesmo vértice. Mesmo assim, o arco gera duas contribuições na soma de todos os graus do grafo, o que mantém o resultado.

O uso de barras para determinar o número de elementos do conjunto E (arestas) pode causar confusão com a representação da função módulo. No contexto a confusão é fácil de evitar, pois E é um conjunto e não um tipo escalar. Em português, a definição do grafo poderia utilizar a letra A para representar o conjunto de arestas. O uso da letra E possivelmente deriva do original em inglês, *edge*.

REFERÊNCIA

CORMEN, Thomas H. *et al.* *Introduction to algorithms*. 3. ed. Cambridge: The MIT Press, 2009.

QUESTÃO 21

No desenvolvimento de um software que analisa bases de DNA, representadas pelas letras A, C, G, T, utilizou-se as estruturas de dados: pilha e fila. Considere que, se uma sequência representa uma pilha, o topo é o elemento mais à esquerda; e se uma sequência representa uma fila, a sua frente é o elemento mais à esquerda.

Analise o seguinte cenário: “a sequência inicial ficou armazenada na primeira estrutura de dados na seguinte ordem: (A,G,T,C,A,G,T,T). Cada elemento foi retirado da primeira estrutura de dados e inserido na segunda estrutura de dados, e a sequência ficou armazenada na seguinte ordem: (T,T,G,A,C,T,G,A). Finalmente, cada elemento foi retirado da segunda estrutura de dados e inserido na terceira estrutura de dados e a sequência ficou armazenada na seguinte ordem: (T,T,G,A,C,T,G,A)”.

Qual a única sequência de estruturas de dados apresentadas a seguir pode ter sido usada no cenário descrito acima?

- A. Fila - Pilha - Fila.
- B. Fila - Fila - Pilha.
- C. Fila - Pilha - Pilha.
- D. Pilha - Fila - Pilha.
- E. Pilha - Pilha - Pilha.

Resposta: alternativa (A)

Autora: Isabel Harb Manssour

COMENTÁRIO

A partir do enunciado da questão é possível saber que:

- um software analisa bases de DNA;
- as bases de DNA são representadas pelas letras A, C, G, T;
- as estruturas de dados utilizadas são pilha e fila;

- a pilha é representada da seguinte maneira:

Topo					
------	--	--	--	--	--
- a fila é representada da seguinte maneira:

Frente					
--------	--	--	--	--	--

O enunciado da questão também descreve o seguinte cenário a ser analisado:

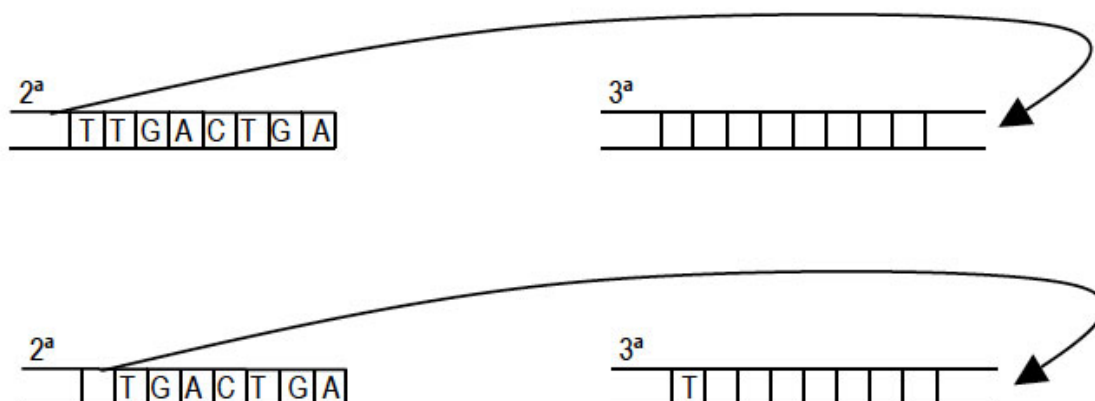
- sequência armazenada na primeira estrutura de dados: A, G, T, C, A, G, T, T;
- elementos retirados da primeira estrutura de dados e armazenados na segunda estrutura de dados na seguinte ordem: T, T, G, A, C, T, G, A;
- elementos retirados da segunda estrutura de dados e armazenados na terceira estrutura de dados na seguinte ordem: T, T, G, A, C, T, G, A.

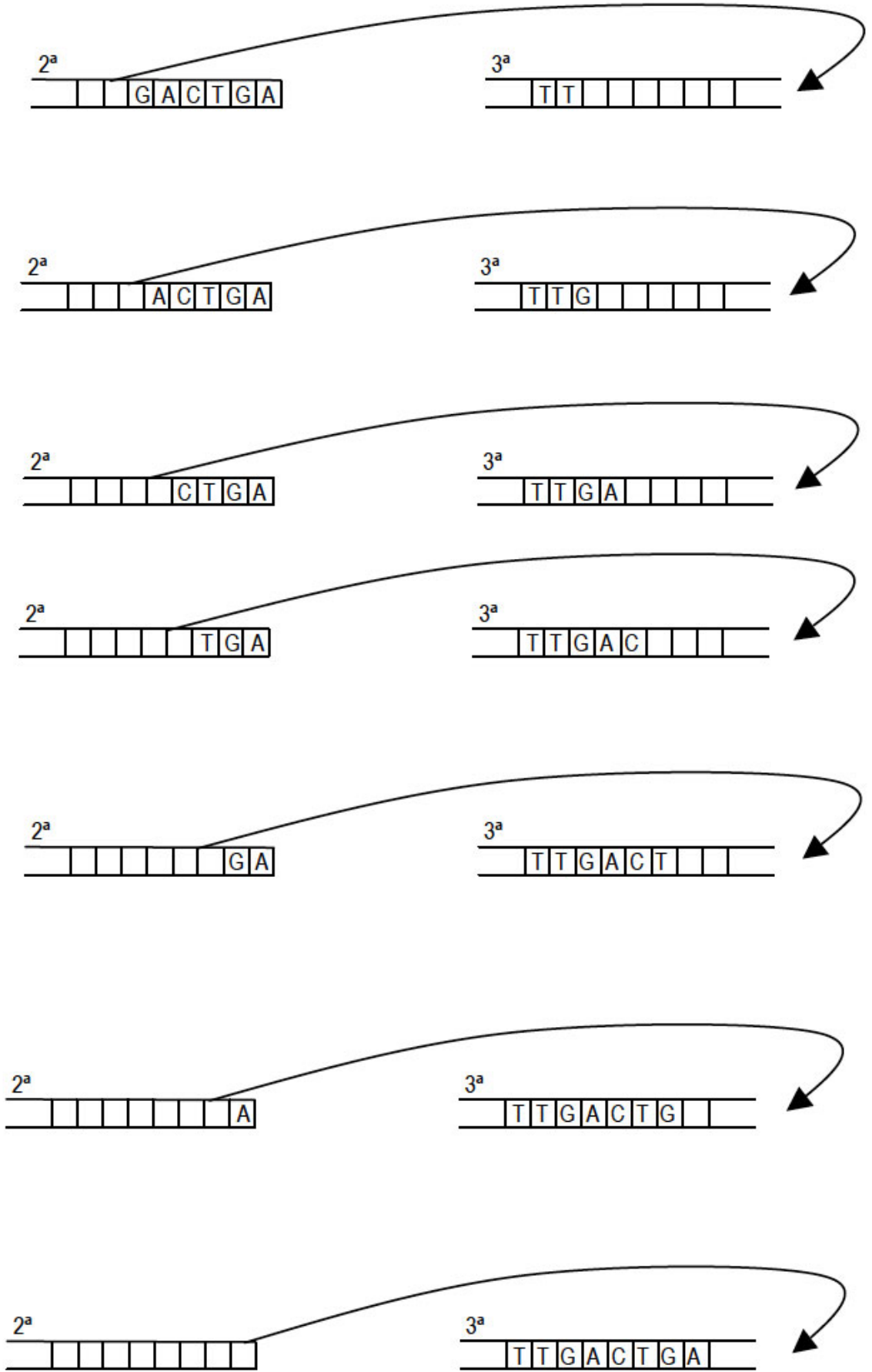
Considerando as definições apresentadas por Goodrich e Tamassia (2007) e Mcallister (2009), sabe-se que:

- Uma **Pilha** consiste em uma coleção de objetos que são inseridos e removidos em um único extremo da estrutura (chamado topo), de acordo com o princípio de acesso aos dados conhecido por LIFO (*Last-In First-Out*, ou seja, último a entrar primeiro a sair). Assim, os elementos são retirados na ordem inversa de sua entrada.
- Uma estrutura de dados do tipo **Fila** também consiste em uma coleção de objetos, porém os mesmos são inseridos e removidos de acordo com o princípio de acesso aos dados conhecido por FIFO (*First-In First-Out*, ou seja, primeiro a entrar primeiro a sair). Assim, os elementos são retirados na mesma ordem de sua entrada.

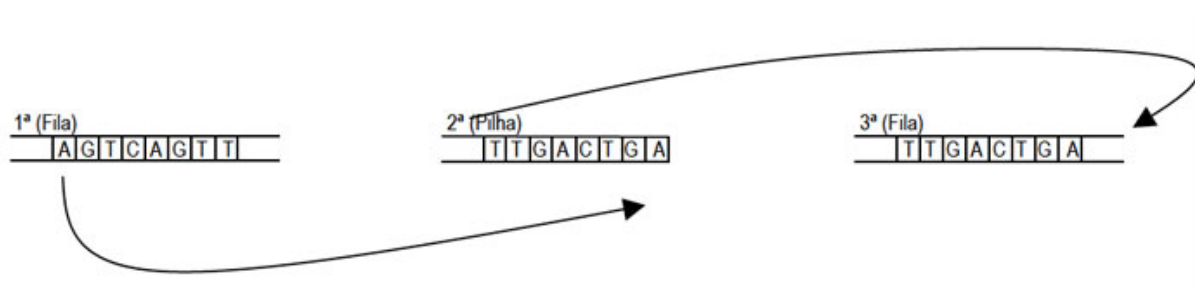
No cenário a ser analisado, observa-se que ao passar da primeira para a segunda estrutura de dados, os elementos ficam na ordem inversa, o que é possível, conforme descrito, se a segunda estrutura de dados for uma pilha.

Por outro lado, ao passar da segunda para a terceira estrutura de dados, a ordem dos dados é mantida, o que é possível se a terceira estrutura de dados for uma fila, como exemplificado passo a passo a seguir.





Sendo assim, considerando as alternativas, a resposta correta é a alternativa A.



REFERÊNCIAS

GOODRICH, Michael T.; TAMASSIA, Roberto. *Estruturas de dados e algoritmos em Java*. 4. ed. Porto Alegre: Bookman, 2007. 600 p.

MCALLISTER, William. *Data Structures and Algorithms Using Java*. 1. ed. Boston: Jones and Bartlett, 2009. 580 p.

QUESTÃO 22

Considere a seguinte tabela verdade, na qual estão definidas quatro entradas – A, B, C e D – e uma saída S.

A	B	C	D	S
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

A menor expressão de chaveamento representada por uma soma de produtos correspondente à saída S é

- A. $AB'(D+C') + A'D' + ABC$.
- B. $AD + A'BD' + A'BC + A'B'C'$.
- C. $A'D' + AB'D + AB'C' + ABC$.
- D. $(A'+D)(A+B+C')(A+B'+C+D')$.
- E. $(A+D')(A'+B'+C)(A'+B+C'+D)$.

Resposta: alternativa (C)

Autor: Fabiano Hessel

COMENTÁRIO

A questão trata de minimização de funções booleanas, cujo resultado deve ser expresso através de uma soma de produtos. Como se trata de soma de produtos, a função do resultado deverá ser composta por mintermos, nos quais as variáveis que possuem valor zero (0) são representadas negadas e as variáveis com valor um (1) são representadas afirmadas. Para fins de solução da questão, nos interessam todos os mintermos nos quais a função responde a um (1) como resultado de saída.

Para solucionar essa questão, podemos utilizar qualquer método de minimização de funções booleanas, como, por exemplo, os teoremas da álgebra de Boole ou mapas de Karnaugh. Neste caso, a maneira mais simples e rápida para determinar a expressão mínima que representa a saída S é a utilização dos mapas de Karnaugh.

Mapa de Karnaugh é uma ferramenta de auxílio à minimização de funções booleanas que utiliza a tabela verdade dessa função como base para as simplificações. Cabe ressaltar que essa ferramenta é bastante útil para funções com até seis (6) variáveis de entrada. Funções com mais de seis (6) variáveis tornam a utilização do mapa muito complexa. Para estes casos devemos utilizar soluções algorítmicas computacionais. Informações complementares a respeito de minimização de funções lógicas e sobre mapas de Karnaugh podem ser encontradas nas referências indicadas abaixo.

Assumindo que o leitor tenha conhecimento prévio sobre como montar e utilizar mapas de Karnaugh, o mapa para a tabela verdade da questão é apresentado na Figura 1. Lembrando que irão nos interessar as saídas nas quais a função responde um (1).

CD \ AB	00	01	11	10
00	1	0	0	1
01	1	0	0	1
11	0	0	1	1
10	1	1	1	0

Figura 1. Mapa de Karnaugh relativo à tabela verdade da questão.

Após construir o mapa, a próxima tarefa é encontrar os mintermos que serão utilizados para construir a expressão final correspondente à saída S. Esses termos são encontrados através do agrupamento de 1s adjacentes no mapa. É importante lembrar que o agrupamento deve ser retangular e deve possuir uma área igual a uma potência de 2. Os retângulos devem ser os maiores possíveis, sem conter nenhum zero (0).

A Figura 2 apresenta os agrupamentos para o mapa da questão.

AB \ CD	00	01	11	10
00	1	0	0	1
01	1	0	0	1
11	0	0	1	1
10	1	1	1	0

Figura 2. Possíveis agrupamentos para o mapa de Karnaugh da questão.

Mintermos relativos aos agrupamentos da Figura 2:

$$A'D' + AB'C' + AB'D + ABC$$

REFERÊNCIAS

FLOYD, THOMAS L. *Sistemas digitais: fundamentos e aplicações*. 9. ed. Porto Alegre: Bookman, 2007. 888 p.

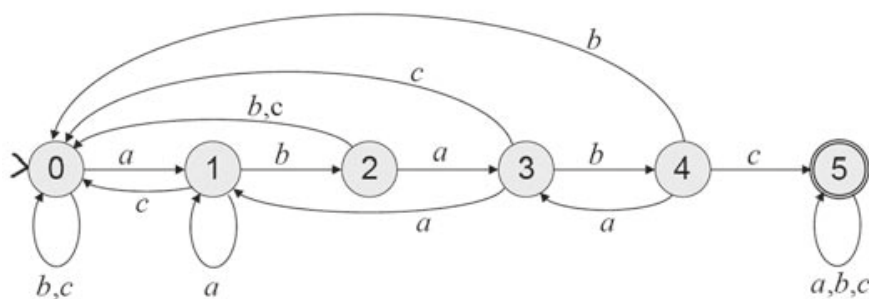
MONTEIRO, M. A. *Introdução à Organização de Computadores*. 5. ed. Rio de Janeiro: LTC, 2012.

STALLINGS, W. *Arquitetura e Organização de Computadores*. 8. ed. São Paulo: Pearson, 2010.

TANENBAUM, A. S. *Organização Estruturada de Computadores*. 5. ed. São Paulo: Pearson, 2007.

QUESTÃO 23

Autômatos finitos possuem diversas aplicações práticas, como na detecção de sequências de caracteres em um texto. A figura abaixo apresenta um autômato que reconhece sequências sobre o alfabeto $\Sigma = \{a, b, c\}$ e uma gramática livre de contexto que gera um Subconjunto de Σ^* , em que λ representa o string vazio.



$$S \rightarrow aS | bS | cS | abA$$

$$A \rightarrow abA | abcB$$

$$B \rightarrow aB | bB | cB | \lambda$$

Analisando a gramática e o autômato acima, conclui-se que

- A. a linguagem gerada pela gramática é inerentemente ambígua.
- B. a gramática é regular e gera uma linguagem livre de contexto.
- C. a linguagem reconhecida pelo autômato é a mesma gerada pela gramática.
- D. o autômato reconhece a linguagem sobre Σ em que os *strings* possuem o prefixo *ababc*.
- E. a linguagem reconhecida pelo autômato é a mesma que a representada pela expressão regular $(a + b + c)^*(ab)^*abc(a + b + c)^*$

Resposta: alternativa (C)

Autor: Alexandre Agustini

COMENTÁRIO

O *Relatório Síntese 2011: Computação* apresenta o seguinte comentário: “[...] Além destas duas, as demais questões com índice fraco de discriminação, questões 17, 23 e 24 também não foram utilizadas no cômputo final das notas, num total de cinco questões eliminadas” (p. 63). Ou seja, a Questão 23 foi considerada muito difícil e, por não possuir índice de discriminação, foi eliminada do cálculo da nota final do Enade 2011.

A questão apresenta um autômato finito e uma gramática para análise por parte dos estudantes. O primeiro desafio apresentado é o entendimento da linguagem gerada por essas duas representações.

Iniciemos analisando a gramática. O símbolo inicial S apresenta três produções recursivas (aS , bS , cS), donde se conclui que a linguagem aceita cadeias iniciais no formato $(a+b+c)^*$. O caso-base dessa recursão é a produção abA , ou seja, a linguagem gerada passa a ter o formato $(a+b+c)^*ab$. A produção A apresenta uma regra recursiva abA e uma produção não recursiva (caso-base) $abcB$. Com isso, podemos reescrever a linguagem gerada na forma: $(a+b+c)^*ab(ab)^*abc$. A produção B , finalmente, possui uma recursão semelhante à primeira produção (aB , bB , cB), ou seja, também gera a cadeia $(a+b+c)^*$, com o caso-base sendo a produção vazia. Dessa forma, temos que a linguagem reconhecida pela gramática é $(a+b+c)^*ab(ab)^*abc(a+b+c)^*$, e esta pode ser reescrita como $(a+b+c)^*(ab)^*ababc(a+b+c)^*$, e, finalmente, como $(a+b+c)^*ababc(a+b+c)^*$, ou seja, a gramática reconhece a linguagem que contém a *substring* $ababc$.

A análise do autômato é um pouco mais complexa, mas uma leitura inicial nos indica que, para realizar a transição do estado inicial 0 ao final 5, é necessário, em algum momento, reconhecer a *substring* β : “ $ababc$ ” (transições 0à1, 1à2, ..., 4à5). A questão importante é se isso é verdadeiro e, também, qual o sufixo dessa *substring*. Tentemos analisar cada um dos estados:

- no estado 0, é possível reconhecer $(b+c)^*$, até que seja realizada uma transição para o estado 1, com o símbolo a (que é o primeiro símbolo de β);
- o estado 1 reconhece a^* e apresenta duas transições para outros estados: no caso de um c volta ao estado 0, onde nenhum símbolo de β foi reconhecido, e com b vai para o estado 2 (dois símbolos de β já reconhecidos);
- o estado 2 apresenta duas possibilidades: com a vai para o estado 3 (três símbolos de β já reconhecidos: aba), ou, com b ou c retorna para o estado inicial, isto é, nenhum símbolo de β foi reconhecido;
- os estados 3 e 4 seguem a mesma estrutura, reconhecendo, cada um a seu tempo, os restantes símbolos de β ;
- as transições do estado 5, finalmente, são óbvias e reconhecem o sufixo $(a+b+c)^*$.

É importante notar que nos estados 0 a 4 todos os símbolos da entrada são sempre válidos, e, também, a única forma de atingir o estado final é passando pela cadeia β . Dessa forma, podemos concluir que a linguagem aceita pelo autômato é $(a+b+c)^*ababc(a+b+c)^*$, a mesma da gramática.

Esta conclusão sobre a linguagem aceita por cada formalismo explica a resposta correta: alternativa **C**.

Segue uma análise do porquê de as outras alternativas serem falsas.

A alternativa **A** coloca que a linguagem gerada pela gramática é inerentemente ambígua. Uma linguagem é inerentemente ambígua se não há gramática não ambígua para esta linguagem. Demonstrar

que a gramática apresentada na questão é ambígua é bastante simples, basta selecionar a cadeia *ababcababc* e demonstrar que a gramática possui duas árvores de derivação para esta entrada, dadas, respectivamente, pelas derivações:

$$S \Rightarrow aS \Rightarrow abS \Rightarrow abaS \Rightarrow ababS \Rightarrow ababcS \Rightarrow ababcbA \Rightarrow ababcbabcB \Rightarrow ababcbabc$$

$$S \Rightarrow a b A \Rightarrow a b a b c B \Rightarrow a b a b c a B \Rightarrow a b a b c a b B \Rightarrow a b a b c a b a B \Rightarrow a b a b c a b a b B \Rightarrow ababcbabcB \Rightarrow ababcbabc$$

Provar que não existe gramática não ambígua para a linguagem é muito difícil, mas basta apresentarmos uma gramática não ambígua para provar que a linguagem não é inerentemente ambígua. E aqui o autor da questão deixou uma dica implícita. Se convertermos o autômato apresentado na questão, será obtida uma gramática não ambígua para a linguagem, logo, a alternativa **A** é falsa.

Utilizando os não terminais “A” a “F”, representando, respectivamente, os estados “0” a “5” do autômato temos a gramática não ambígua:

$$\begin{aligned} A &\rightarrow bA \mid cA \mid aB \\ B &\rightarrow aB \mid cA \mid bC \\ C &\rightarrow bA \mid cA \mid aD \\ D &\rightarrow cA \mid aB \mid bE \\ E &\rightarrow aD \mid bA \mid cF \\ F &\rightarrow aF \mid bF \mid cF \mid \lambda \end{aligned}$$

A alternativa “B” é falsa, além de a gramática apresentada não ser regular, linguagens regulares não geram linguagens livres de contexto.

A alternativa “D” é facilmente demonstrável como falsa pois a cadeia “aaababc” é reconhecida pelo autômato e não contém o prefixo “ababc”.

A alternativa “E” é, também, facilmente demonstrável como falsa pois a expressão apresentada gera a cadeia “abc”, que não é reconhecida nem pelo autômato nem pela linguagem.

REFERÊNCIAS

AHO, A. V.; SETHI, S.; ULLMAN, J. D. *Compiladores: princípios, técnicas e ferramentas*. 2. ed. São Paulo: Pearson, 2008. 634p.

HOPCROFT, John E.; MOTWANI, Rajeev; ULLMAN, Jeffrey D. *Introdução à teoria de autômatos, linguagens e computação*. Rio de Janeiro: Elsevier, 2003. 560 p.

INEP. Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira, Ministério da Educação, Brasil. *Relatório Síntese 2011: Computação 2013*. Disponível em: <<http://portal.inep.gov.br/enade/relatorio-sintese-2011>>. Acesso em: abr. 2014.

QUESTÃO 24

As filas de prioridades (heaps) são estruturas de dados importantes no projeto de algoritmos. Em especial, heaps podem ser utilizados na recuperação de informação em grandes bases de dados constituídos por textos. Basicamente, para se exibir o resultado de uma consulta os documentos recuperados são ordenados de acordo com a relevância presumida para o usuário. Uma consulta pode recuperar milhões de documentos que certamente não serão todos examinados. Na verdade, o usuário examina os primeiros m documentos dos n recuperados, em que m é da ordem de algumas dezenas.

Considerando as características dos heaps e sua aplicação no problema descrito acima, avalie as seguintes afirmações.

- I. Uma vez que o heap é implementado como uma árvore binária de pesquisa essencialmente completa, o custo computacional para sua construção é $O(n \log n)$.
- II. A implementação de heaps utilizando-se vetores é eficiente em tempo de execução e em espaço de armazenamento, pois o pai de um elemento armazenado na posição i se encontra armazenado na posição $2i+1$.
- III. O custo computacional para se recuperar de forma ordenada os m documentos mais relevantes armazenados em um heap de tamanho n é $O(m \log n)$.
- IV. Determinar o documento com maior valor de relevância armazenado em um heap tem custo computacional $O(1)$.

Está correto apenas o que se afirma em

- A. I e II.
- B. II e III.
- C. III e IV.
- D. I, II e IV.
- E. I, III e IV.

Resposta: alternativa (C)

Autor: João Batista Souza de Oliveira

COMENTÁRIO

Vamos analisar o enunciado da questão e as afirmações dadas. O enunciado fala sobre *heaps* sem dar detalhes de estrutura, de implementação ou operação, mas apresenta um caso em que desejamos recuperar os m elementos mais importantes de um *heap* com n elementos totais. Vamos para as afirmações.

A afirmação I informa que *heaps* são implementados como árvores binárias de pesquisa essencialmente completas e, portanto, o custo de construção de um *heap* é $O(n \log n)$.

Análise: em primeiro lugar, *heaps* geralmente não são implementados como árvores binárias de pesquisa. Eles costumam ser modelados como árvores binárias sim, mas não de pesquisa: a regra geralmente imposta à estrutura obriga que o valor do nodo-pai seja maior ou igual aos valores de seus filhos, o que faz os maiores valores estarem próximos à raiz, mas sem preferência por direita ou esquerda.

Em segundo lugar, mesmo sendo modelado como árvore binária, geralmente a estrutura é mapeada dentro de um *array* ou vetor, indexado de forma que o nodo 0 seja a raiz da árvore e para cada nodo na posição i seu filho esquerdo esteja na posição $2i+1$ e o direito esteja na posição $2i+2$. Isso faz com que possamos usar o acesso direto do *array* e torna desnecessário criar a estrutura (crie um vetor de 15 elementos e desenhe a árvore correspondente: você confirmará que ela contém todos os elementos do vetor e é balanceada).

Agora que sabemos que o *heap* estará inserido dentro de um vetor, resta a pergunta de como reorganizar os elementos para que o critério de ordem (pai maior ou igual a filhos) seja respeitado. Para isso existem duas formas: se realmente imitarmos uma árvore binária e fizermos n inserções de elementos na árvore, como a altura dela está limitada a $\log(n)$, o desempenho final será mesmo $O(n \log n)$. No entanto, esta não é a maneira mais eficiente, pois é melhor começar com a árvore organizada de forma arbitrária e construir o *heap* a partir dos níveis inferiores: para cada subárvore, iniciando de baixo, deixamos a raiz da subárvore descer até ser maior do que seus filhos. No início temos várias subárvores pequenas, que vão sendo tratadas aos poucos como subárvores maiores e, com isso, mais elementos competem para subir na estrutura. A análise do desempenho dessa operação pode ser encontrada em vários livros de algoritmos ou na web e é muito interessante. O resultado final é que a montagem de um *heap*, quando feita a partir dos níveis inferiores, é $O(n)$.

Como *heaps* não são implementados como árvores de pesquisa e sua construção pode ser feita em tempo menor do que $O(n \log n)$, a afirmação I está incorreta.

A afirmação II assevera que o pai de um elemento armazenado na posição i estará na posição $2i+1$. Se for assim, existe um problema imediato: todos os elementos na segunda metade do vetor não terão pai, por que $2i+1$ estará fora do vetor. Como uma árvore só pode ter um nodo sem pai, a raiz, esta afirmação está errada. O certo seria dizer que o pai de um nodo na posição i está na posição $\text{floor}((i-1)/2)$ para um vetor indexado a partir de 0.

Na afirmação III argumenta-se que recuperar os m documentos mais relevantes em um *heap* de n elementos custa $O(m \log n)$. Recuperar o documento mais relevante é simples: basta acessar a posição 0 do vetor, o que custa $O(1)$. No entanto, ao retirá-lo do *heap*, um novo elemento deve tomar o seu lugar. Este elemento deve ser o maior entre seus dois filhos, que sobe para ocupar a posição e deixa o seu lugar vago um nível abaixo. Um de seus filhos deve ocupar o espaço e assim sucessivamente até o último nível da árvore, que tem $O(\log n)$ níveis. Só então o *heap* está reconstituído, e fazer esta operação m vezes custa, portanto, $O(m \log n)$. A afirmação está correta.

A afirmação **IV** diz que encontrar o documento mais importante tem custo $O(1)$. A análise da afirmação anterior justifica esta resposta, pois o documento mais relevante terá a maior prioridade de todos e, portanto, tem que estar na raiz do *heap*, ou seja, na posição 0. O acesso é direto, o que custa $O(1)$.

Ao final, apenas as afirmações **III** e **IV** estão corretas, o que leva à alternativa **C**¹.

REFERÊNCIA

CORMEN, Thomas H.; LEISERSON, Charles E.; RIVEST, Ronald L.; STEIN, Clifford. *Introduction to Algorithms*. 3. ed. Cambridge: MIT Press, 2009.

¹ Por outro lado, uma dedução muito elegante do custo da construção de um heap pode ser encontrada em: <https://www.cs.umd.edu/users/meesh/cmsc351/mount/lectures/lect14-heapsort-analysis-part.pdf>

QUESTÃO 25

Um Padrão de Projeto nomeia, abstrai e identifica os aspectos-chave de uma estrutura de projeto comum para torná-la útil para a criação de um projeto orientado a objetos reutilizáveis.

GAMMA, E.; HELM, R.; JOHNSON, R.; VLISSIDES, J.
Padrões de Projeto-Soluções Reutilizáveis de Software Orientado a Objetos.
Porto Alegre: Bookman, 2000.

Em relação a Padrões de Projeto, analise as afirmações a seguir.

- I. Prototype é um tipo de padrão estrutural.
- II. Singleton tem por objetivos garantir que uma classe tenha ao menos uma instância e fornecer um ponto global de acesso para ela.
- III. Template Method tem por objetivo definir o esqueleto de um algoritmo em uma operação, postergando a definição de alguns passos para subclasses.
- IV. Iterator fornece uma maneira de acessar sequencialmente os elementos de um objeto agregado sem expor sua representação subjacente.

É correto apenas o que se afirma em

- A. I.
- B. II.
- C. I e IV.
- D. II e III.
- E. III e IV.

Resposta: alternativa (E)

Autora: Ana Paula Terra Bacelo

COMENTÁRIO

Esta questão trata a respeito dos Padrões de Projeto propostos por Erich Gamma *et al.* (2000). O uso de padrões é uma das técnicas de reutilização de software. Além dessas, podemos considerar o uso e reúso de componentes, linhas de produto de software, dentre outros.

No que se refere a esses padrões de projeto, os autores propuseram um catálogo de padrões oriundos de soluções recorrentes em projetos de desenvolvimento orientado a objetos. Uma das grandes vantagens no uso de padrões de projeto é o reúso de soluções recorrentes e, conseqüentemente, o aumento da qualidade e a redução de custo do software. Esses padrões de projeto são descritos através das seguintes propriedades: intenção, motivação, aplicabilidade, implementação, padrões relacionados e a forma como é conhecido. Os padrões que fazem parte desse catálogo são categorizados como padrões de criação, estrutural e comportamental.

Considerando as afirmativas descritas na questão, pode-se avaliar que:

- a afirmativa **I** está incorreta, pois o Prototype é considerado um padrão de criação no catálogo de Padrões de Projeto de Erich Gamma *et al.* (2000);
- a afirmativa **II** está incorreta, pois o padrão Singleton garante que uma classe tenha somente uma única instância, e não ao menos uma instância como descrito na mesma;
- a afirmativa **III** está correta;
- a afirmativa **IV** está correta.

Sendo assim, está correto afirmar que a alternativa **E** é a afirmativa correta dessa questão.

REFERÊNCIA

GAMMA, E.; HELM, R.; JOHNSON, R. et al. *Padrões de Projeto – Soluções Reutilizáveis de Software Orientado a Objetos*. Porto Alegre: Bookman, 2000. 363p.

QUESTÃO 26



Um baralho tem 52 cartas, organizadas em 4 naipes, com 13 valores diferentes para cada naipe. Os valores possíveis são: Ás, 2, 3, ..., 10, J, Q, K.

No jogo de poker, uma das combinações de 5 cartas mais valiosas é o full house, que é formado por três cartas de mesmo valor e outras duas cartas de mesmo valor. São exemplos de full houses:

- I) três cartas K e duas 10 (como visto na figura) ou
- II) três cartas 4 e duas Ás.

Quantas possibilidades para full house existem em um baralho de 52 cartas?

- A. 156.
- B. 624.
- C. 1872.
- D. 3744.
- E. 7488.

Resposta: alternativa (D)

Autor: Sérgio Kakuta Kato

COMENTÁRIO

Dentro de probabilidade, essa questão se enquadra nos modelos discretos, particularmente na distribuição multinomial. No entanto, devido à natureza da questão (cartas de um baralho), ela é mais facilmente resolvida sem aplicação das fórmulas, e sim de forma intuitiva, uma vez que cada carta do baralho tem 2 classificações distintas (valores e naipe). A forma mais intuitiva e rápida de resolução é apresenta a seguir.

Num baralho de 52 cartas, existem 13 diferentes valores possíveis, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K e Ás, para cada naipe. Num jogo de Poker o *full house* é composto por três cartas do mesmo valor (uma trinca) e duas cartas de mesmo valor (um par).

Existem 13 diferentes pares possíveis. A possibilidade de formar uma trinca se restringe a 12, pois uma das figuras será usada para formar um par. Com isso, existem 156 ($13 \cdot 12$) composições de trinca e dupla.

Considerando como exemplo a composição K9, formada por K, K, K, 9, 9. Como o baralho é composto por 4 naipes, para a obtenção de 3 valores K, existem, então, 4 combinações de 3 a 3 ($\binom{4}{3}$), ou seja, 4 formas diferentes de ter 3 valores K; e existem 4 combinações 2 a 2 ($\binom{4}{2}$), ou seja, 6 formas diferentes de termos dois valores nove.

Essas possibilidades se repetem para cada uma das 156 composições de *full house* distintas, sendo assim as possibilidades para *full house* em um baralho de 52 cartas são:

REFERÊNCIAS

BUSSAB, W. O; MORETTIN, P. A. *Estatística Básica*. 5. ed. São Paulo: Saraiva, 2002.

MEYER, Paul L. *Probabilidade: aplicações à estatística*. Rio de Janeiro: LTC, 1987.

QUESTÃO 27

Um dos problemas clássicos da computação científica é a multiplicação de matrizes. Assuma que foram declaradas e inicializadas três matrizes quadradas de ponto flutuante, a , b e c , cujos índices variam entre 0 e $n - 1$. O seguinte trecho de código pode ser usado para multiplicar matrizes de forma sequencial:

```
1. for [i = 0 to n - 1] {  
2.   for [j = 0 to n - 1] {  
3.     c[i, j] = 0.0;  
4.     for [k = 0 to n - 1]  
5.       c[i, j] = c[i, j] + a[i, k] * b[k, j];  
6.   }  
7. }
```

O objetivo é paralelizar esse código para que o tempo de execução seja reduzido em uma máquina com múltiplos processadores e memória compartilhada. Suponha que o comando “co” seja usado para definição de comandos concorrentes, da seguinte forma: “co [i = 0 to n - 1] { x; y; z; }” cria n processos concorrentes, cada um executando sequencialmente uma instância dos comandos x , y , z contidos no bloco.

Avalie as seguintes afirmações sobre o problema.

- I. Esse problema é exemplo do que se chama “embaraçosamente paralelo”, porque pode ser decomposto em um conjunto de várias operações menores que podem ser executadas independentemente.
- II. O programa produziria resultados corretos e em tempo menor do que o sequencial, trocando-se o “for” na linha 1 por um “co”.
- III. O programa produziria resultados corretos e em tempo menor do que o sequencial, trocando-se o “for” na linha 2 por um “co”.
- IV. O programa produziria resultados corretos e em tempo menor do que o sequencial, trocando-se ambos “for”, nas linhas 1 e 2, por “co”.

É correto o que se afirma em

- A. I, II e III, apenas.
- B. I, II e IV, apenas.
- C. I, III e IV, apenas.
- D. II, III e IV, apenas.
- E. I, II, III, IV.

Resposta: alternativa (E)

Autor: Tiago Ferreto

COMENTÁRIO

A questão apresenta um trecho de código sequencial para calcular a multiplicação de matrizes quadradas com dimensões $n \times n$. O comando “co” é apresentado para execução das iterações do laço de repetição “for” de maneira concorrente, considerando o uso de uma máquina multiprocessada e com memória compartilhada. O comando “co” é análogo ao uso do comando “omp parallel for” do padrão OpenMP. Após a apresentação do problema, a questão pede que um conjunto de afirmações seja avaliada.

A afirmação I indica que o problema pertence à classe dos problemas “embaraçosamente paralelos” (*Embarrassingly Parallel*). Esses problemas são caracterizados pela facilidade em dividi-los em tarefas independentes, devido à inexistência de dependência entre as tarefas. No problema apresentado, uma forma direta de paralelizar o trecho de código consiste em substituir o primeiro “for” (linha 1) pelo comando “co”. Para garantir o funcionamento correto da versão paralela do problema, deve ser garantida a inexistência de dependências entre as tarefas paralelas (iteraões do “for”). Analisando o trecho de código, é possível verificar que cada iteração utilizará um valor diferente na variável i que é utilizado para indexar as matrizes a e c . Dessa forma, cada iteração irá acessar áreas de memória distintas, com exceção da matriz b , que é compartilhada entre as tarefas. Considerando que a matriz b permanece constante durante toda a execução, conclui-se que não existe dependência entre as tarefas e o problema pode ser facilmente paralelizado, sendo considerado como “embaraçosamente paralelo”. Portanto, a afirmativa I é verdadeira.

A afirmação II indica que o programa produziria resultados corretos e em tempo menor do que o sequencial, trocando-se o “for” na linha 1 por um “co”. De acordo com a análise realizada na afirmação I, verificou-se que a alteração do “for” da linha 1 por um “co” pode ser utilizada, pois não existe dependência entre as tarefas. Considerando que cada iteração do “for” da linha 1 leve um tempo T , a versão sequencial levaria um tempo $n * T$, e a versão paralela levaria um tempo T utilizando n processadores. Logo, o tempo seria menor que o sequencial, e a afirmação II é verdadeira.

A afirmação III indica que o programa produziria resultados corretos e em tempo menor do que o sequencial, trocando-se o “for” na linha 2 por um “co”. Utilizando o mesmo tipo de análise realizada para a afirmação I, deve ser verificado se não existem áreas de memória comuns que sejam alteradas pelas iterações durante a sua execução em paralelo. Nesse caso, cada iteração utilizará um valor distinto para a variável j , que é utilizada para indexar as matrizes c e b . Dessa forma, cada iteração irá acessar áreas de memória distintas, com exceção da matriz a , que é compartilhada entre as tarefas. Considerando que a matriz a permanece constante durante toda a execução, conclui-se que não existe dependência entre as tarefas, e o programa irá produzir resultados corretos. Utilizando a mesma análise realizada para a afirmação II, se for considerado que cada iteração do “for” da linha 2 leve um tempo T , a versão sequencial (do laço da linha 2) levaria um tempo $n * T$, e a versão paralela levaria um tempo T utilizando n processadores. Portanto, o tempo da versão paralela seria menor que a versão sequencial, e a afirmação III é verdadeira.

A afirmação IV indica que o programa produziria resultados corretos e em tempo menor do que o sequencial, trocando-se ambos os “for”, nas linhas 1 e 2, por “co”. Baseando-se nas análises realizadas

nas afirmações anteriores, verificou-se que não há dependência entre as iterações decorrentes do uso do comando “*co*” nas linhas 1 e 2. Dessa forma, ambos “*for*” podem ser substituídos produzindo resultados corretos. Considerando que cada iteração do “*for*” da linha 2 leve um tempo T , estima-se que o tempo total do trecho de código sequencial seria $n * n * T$. Se fossem utilizados $n * n$ processadores, o tempo paralelo estimado seria T . Logo, o tempo é menor que o sequencial, e a afirmação **IV** é verdadeira.

Considerando que as afirmações **I**, **II**, **III** e **IV** **são** verdadeiras, então a resposta correta é a alternativa **E**.

REFERÊNCIAS

GRAMA, A. et al. *Introduction to parallel computing*. 2. ed. Harlow: Pearson Education, 2003. 636 p.

QUINN, M.J. *Parallel Programming in C with MPI and OpenMP*. Boston: McGraw Hill, 2004. 529 p.

QUESTÃO 28

Algoritmos criados para resolver um mesmo problema podem diferir de forma drástica quanto a sua eficiência. Para evitar este fato, são utilizadas técnicas algorítmicas, isto é, conjunto de técnicas que compreendem os métodos de codificação de algoritmos de forma a salientar sua complexidade, levando-se em conta a forma pela qual determinado algoritmo chega à solução desejada.

Considerando os diferentes paradigmas e técnicas de projeto de algoritmos, analise as afirmações abaixo.

- I. A técnica de tentativa e erro (backtracking) efetua uma escolha ótima local, na esperança de obter uma solução ótima global.
- II. A técnica de divisão e conquista pode ser dividida em três etapas: dividir a instância do problema em duas ou mais instâncias menores; resolver as instâncias menores recursivamente; obter a solução para as instâncias originais (maiores) por meio da combinação dessas soluções.
- III. A técnica de programação dinâmica decompõe o processo em um número finito de subtarefas parciais que devem ser exploradas exaustivamente.
- IV. O uso de heurísticas (ou algoritmos aproximados) é caracterizado pela ação de um procedimento chamar a si próprio, direta ou indiretamente.

É correto apenas o que se afirma em

- A. I.
- B. II.
- C. I e IV.
- D. II e III.
- E. III e IV.

Resposta: alternativa (B)

Autor: João Batista Souza de Oliveira

COMENTÁRIO

Vamos analisar cada afirmativa a seguir.

- I. *Backtracking* explora todo o espaço de busca, rejeitando partes dele quando determina que a solução desejada não pode estar na parte examinada. Não é obrigatório que a escolha da região a examinar seja feita de alguma maneira especial, pois o processo geralmente vasculha todo o espaço de qualquer maneira. No entanto, a frase que fala de uma técnica que faz escolhas ótimas locais esperando encontrar um ótimo global é praticamente uma descrição de como funcionam os algoritmos gulosos.
- II. Esta é uma descrição curta porém muito acertada da técnica de divisão e conquista. O problema deve ser dividido em instâncias menores, estas devem ser resolvidas e a resposta do problema original deve ser reconstituída com as respostas das instâncias menores. A afirmação ainda diz que o processo de solução é recursivo, o que é geralmente verdade, mas não é forçoso que seja assim.
- III. Programação dinâmica geralmente envolve a recursão e a quebra de problemas em instâncias menores que devem ser resolvidas, mas também inclui o uso de técnicas de aceleração como memorização, que servem exatamente para evitar que tenhamos de explorar exaustivamente as subtarefas parciais, reutilizando seus resultados.
- IV. Heurística é a tentativa de aproximação de algoritmos a uma resposta desejada, porém sem garantia de que irá encontrá-la exatamente e é usada quando um algoritmo que fornece essa garantia é muito caro para uso prático. No entanto, não existe nenhuma obrigação de uma heurística envolver um procedimento que chame a si mesmo (ou seja, use recursão).

Em consequência, a única resposta possível é a alternativa **B**, que aceita como verdadeira apenas a afirmação **II**, sobre divisão e conquista.

REFERÊNCIA

HOROWITZ, Ellis; SAHNI, Sartaj; RAJASEKARAN, Sanguthevar. *Computer Algorithms*. Summit: Silicon Press, 2008.

QUESTÃO 29

Uma antiga empresa de desenvolvimento de software resolveu atualizar toda sua infraestrutura computacional adquirindo um sistema operacional multitarefa, processadores multi-core (múltiplos núcleos) e o uso de uma linguagem de programação com suporte a threads.

O sistema operacional multitarefa de um computador é capaz de executar vários processos (programas) em paralelo. Considerando esses processos implementados com mais de uma thread (multi-threads), analise as afirmações abaixo.

- I. Os ciclos de vida de processos e threads são idênticos.
- II. Threads de diferentes processos compartilham memória.
- III. Somente processadores multi-core são capazes de executar programas multi-threads.
- IV. Em sistemas operacionais multitarefa, threads podem migrar de um processo para outro.

É correto apenas o que se afirma em

- A. I.
- B. II.
- C. I e III.
- D. I e IV.
- E. II e IV.

Resposta: alternativa (A)

Autor: César Augusto Fonticielha De Rose

COMENTÁRIO

Com a proliferação de arquiteturas com vários núcleos (*multi-core*), uma das técnicas para acelerar a execução de um **único** processo nessas máquinas é subdividi-lo em várias linhas de execução independentes, chamadas de *threads* ou processos leves. Dessa forma o escalonador do sistema operacional pode delegar essas *threads* para núcleos diferentes do sistema, explorando o paralelismo da arquitetura no intuito de reduzir o tempo de execução do processo original.

A única afirmação correta das 4 apresentadas é a afirmação I, pois *threads* possuem o mesmo ciclo de vida que processos pesados (são criadas, executam e terminam), passando pelos mesmos estados (prontas para executar, bloqueadas, terminadas etc.).

Threads compartilham memória apenas quando pertencem ao mesmo processo, de forma que a afirmação II **não está correta**.

Programas compostos de várias *threads* podem executar também em arquiteturas que não sejam *multicore*, neste caso concorrendo pelos recursos existentes, o que torna a afirmação III incorreta. A tendência neste caso é que demorem mais para executar devido à menor exploração do potencial paralelismo entre elas.

Threads podem migrar entre processadores/núcleos de uma máquina durante a execução, mas durante seu ciclo de vida sempre pertencem ao mesmo processo, de forma que a afirmação IV **não está correta**.

REFERÊNCIAS

DE ROSE, Cesar A. F.; NAVAUX, Philippe O. A. *Arquiteturas paralelas*. Porto Alegre: Sagra Luzzatto, 2003. 152 p. (Série Livros Didáticos; 15)

SILBERSCHATZ, Abraham; GALVIN, Peter B.; GAGNE, Greg. *Operating system concepts*. 7. ed. John Wiley & sons, 2004. 887p.

TANENBAUM, A. S.; WOODHULL, A. S. *Sistemas Operacionais: Projeto e Implementação*. 3. ed. Porto Alegre: Bookman, 2008. 992p.

QUESTÃO 30

Suponha que se queira pesquisar a chave 287 em uma árvore binária de pesquisa com chaves entre 1 e 1 000. Durante uma pesquisa como essa, uma sequência de chaves é examinada.

Cada sequência abaixo é uma suposta sequência de chaves examinadas em uma busca da chave 287.

- I. 7, 342, 199, 201, 310, 258, 287
- II. 110, 132, 133, 156, 289, 288, 287
- III. 252, 266, 271, 294, 295, 289, 287
- IV. 715, 112, 530, 249, 406, 234, 287

É válido apenas o que se apresenta em

- A. I.
- B. III.
- C. I e II.
- D. II e IV.
- E. III e IV.

Resposta: alternativa (C)

Autor: Michael da Costa Móra

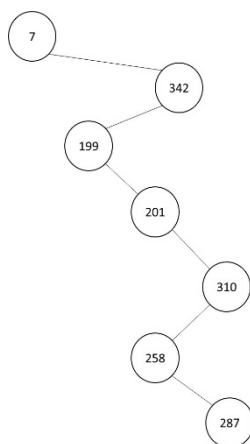
COMENTÁRIO

Uma Árvore Binária de Pesquisa (ABP) é uma árvore binária na qual, para cada nó x da árvore, se y é um nó na subárvore da esquerda de x , então $y < x$, e se y é um nó na subárvore da direita de x , então $x \leq y$ (CORMEN, 2009). Ou seja, para qualquer nó da árvore, os elementos à sua esquerda são menores do que este, e os à sua direita são maiores ou iguais àquele nó. Isso permite que a busca de uma chave nessa árvore seja otimizada, uma vez que não é necessário percorrer toda a árvore, mas a cada nó testar se a chave desejada é menor ou maior que a chave corrente, e prosseguir a busca na subárvore adequada.

Uma consequência dessa propriedade é que a sequência de busca (ou seja, a sequência de chaves testadas em busca da chave desejada) deve gerar uma árvore parcial que respeita essa propriedade. Por exemplo, na sequência

I. 7, 342, 199, 201, 310, 258, 287

a árvore parcial gerada seria

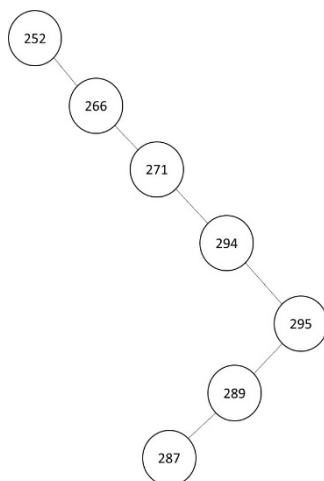


e, para cada nodo da árvore, os elementos na subárvore da esquerda são menores que o nodo, e os elementos na subárvore da direita são maiores ou iguais ao nodo. O mesmo acontece na sequência II.

Já na sequência:

III. 252, 266, 271, 294, 295, 289, 287

a árvore parcial gerada seria



colocando as chaves 287 e 289 na subárvore da direita da chave 294, violando a propriedade que define uma ABP. Também, na sequência IV, a propriedade é violada, pois a chave 234 fica colocada na subárvore da direita da chave 249. Assim, as sequências válidas são as apresentadas em I e II. Logo, a resposta correta é a alternativa **C**.

REFERÊNCIA

CORMEN, T.H.; LEISERSON, C.E.; RIVEST, R.L.; STEIN, C. *Introduction to Algorithms*. Boston (MA): The MIT Press, 2009.

QUESTÃO 31

Na Sociologia da Educação, o currículo é considerado um mecanismo por meio do qual a escola define o plano educativo para a consecução do projeto global de educação de uma sociedade, realizando, assim, sua função social. Considerando o currículo na perspectiva crítica da Educação, avalie as afirmações a seguir.

- I. O currículo é um fenômeno escolar que se desdobra em uma prática pedagógica expressa por determinações do contexto da escola.
- II. O currículo reflete uma proposta educacional que inclui o estabelecimento da relação entre o ensino e a pesquisa, na perspectiva do desenvolvimento profissional docente.
- III. O currículo é uma realidade objetiva que inviabiliza intervenções, uma vez que o conteúdo é condição lógica do ensino.
- IV. O currículo é a expressão da harmonia de valores dominantes inerentes ao processo educativo.

É correto apenas o que se afirma em

- A. I.
- B. II.
- C. I e III.
- D. II e IV.
- E. III e IV.

Resposta: alternativa (B)

Autora: Zuleica Almeida Rangel

COMENTÁRIO

A afirmação I é considerada incorreta na medida em que se distancia do que se entende por currículo numa perspectiva crítica de Educação. Ao se conceber currículo como trajetória, ação e caminhada construída de forma coletiva e diferenciada, respeitando cada realidade escolar, também o entendemos como processo dinâmico, aberto e flexível. Assim, o currículo passa a ser visto não mais como “um fenômeno escolar que se desdobra em uma prática pedagógica expressa por determinações do contexto da escola”, e sim como um processo mutante, sujeito a inúmeras influências, que privilegia o

conhecimento acumulado pela humanidade como ponto de partida, buscando-se as mediações entre esse conhecimento e a realidade do aluno.

A afirmação II é correta, pois aborda a relação entre o ensino e a pesquisa como aspecto a ser expresso nas propostas educacionais que os currículos abarcam, uma vez que é imprescindível a ação de educadores e de educandos com a realidade, fundamentada no trabalho com o conhecimento elaborado, a fim de possibilitar “o compreender, o usufruir ou o transformar a realidade” (VASCONCELLOS, 2012, p. 98). A partir dessa ideia, conforme a afirmação, a relação entre o ensino e a pesquisa, na perspectiva do desenvolvimento profissional docente, é o que garante ao professor uma prática que se estende à perspectiva da formação do aluno, considerando-se, como nos diz Pedro Demo, que “o cerne mais palpável da competência está na pesquisa, compreendida não só como expediente de construção científica, mas igualmente como processo formativo” (1995, p. 53). Ademais, um currículo, na perspectiva crítica da Educação, deve ser um processo que tenha como objetivos levar o aluno a pensar por si mesmo e o educador a tomar parte do processo de aquisição do conhecimento.

A afirmação III é incorreta, uma vez que o currículo é uma prática dialógica entre agentes sociais, educandos e educadores, que, juntos, concretizam as funções da escola numa maneira particular de enfocar os conteúdos de ensino, num determinado momento histórico e social. O currículo procura responder a algumas perguntas fundamentais: o que ensinar? Quando ensinar? Como ensinar? O que, quando e como avaliar? Com quem planejar, ensinar e avaliar? Dessa forma, não pode “ser pensado apenas como um rol de conteúdos a serem transmitidos para um sujeito passivo [...]”. Nesse sentido, o currículo que nos interessa é aquele em que o educando tem oportunidade de entrar no movimento do conceito” (VASCONCELLOS, 2012, p. 99).

A afirmação IV é incorreta, pois nega a expressão da função socializadora e cultural do currículo da escola, que é a de assegurar a seus membros a aquisição da experiência social historicamente acumulada e socialmente organizada pela humanidade.

REFERÊNCIA

VASCONCELLOS, Celso dos Santos. *Planejamento – Projeto de Ensino-Aprendizagem e Projeto Político-Pedagógico*. Elementos metodológicos para elaboração e realização. 22. ed. São Paulo: Libertad Editora, 2012.

QUESTÃO 32

O fazer docente pressupõe a realização de um conjunto de operações didáticas coordenadas entre si. São o planejamento, a direção do ensino e da aprendizagem e a avaliação, cada uma delas desdobradas em tarefas ou funções didáticas, mas que convergem para a realização do ensino propriamente dito.

LIBÂNEO, J. C. *Didática*. São Paulo: Cortez, 2004, p. 72.

Considerando que, para desenvolver cada operação didática inerente ao ato de planejar, executar e avaliar, o professor precisa dominar certos conhecimentos didáticos, avalie quais afirmações abaixo se referem a conhecimentos e domínios esperados do professor.

- I. Conhecimento dos conteúdos da disciplina que leciona, bem como capacidade de abordá-los de modo contextualizado.
- II. Domínio das técnicas de elaboração de provas objetivas, por se configurarem instrumentos quantitativos precisos e fidedignos.
- III. Domínio de diferentes métodos e procedimentos de ensino e capacidade de escolhê-los conforme a natureza dos temas a serem tratados e as características dos estudantes.
- IV. Domínio do conteúdo do livro didático adotado, que deve conter todos os conteúdos a serem trabalhados durante o ano letivo.

É correto apenas o que se afirma em

- A. I e II.
- B. I e III.
- C. II e III.
- D. II e IV.
- E. III e IV.

Resposta: alternativa (B)

Autora: Ana Lúcia Souza de Freitas

COMENTÁRIO

A afirmação I está correta, pois pressupõe que o conteúdo específico de uma disciplina é um conhecimento necessário, mas não suficiente para o fazer docente. O que o professor precisa saber para ensinar (LIMA, 2008), ou seja, o conhecimento profissional docente é um tema amplo, que ganha relevância quando considerado na perspectiva da superação da racionalidade técnica. Ao longo do século XX, a racionalidade técnica prevaleceu como referência para a formação docente, assim como para outros profissionais, reduzindo a prática a um espaço de aplicação de conhecimentos acadêmicos. Tal perspectiva de formação desconsiderou as singularidades de cada contexto, bem como a dinamicidade das situações em que a experiência cotidiana requer, juntamente com definições técnicas, opções políticas e éticas (GÓMEZ, 1997).

A afirmação II está incorreta, pois utiliza o conhecimento técnico a respeito da elaboração de provas objetivas para justificar a infalibilidade desse tipo de instrumento, desconsiderando as contribuições do conhecimento pedagógico acerca das limitações do uso exclusivo de provas, sugerindo que essas sejam associadas a outros tipos de instrumento para a obtenção de dados para a avaliação da aprendizagem (GRILLO; GESSINGER, 2010).

A afirmação III está correta, pois traz implícita uma concepção ampliada do fazer docente, que inclui o conhecimento pedagógico como um dos componentes relevantes do conhecimento profissional docente (LIMA, 2008).

A afirmação IV está incorreta, visto que apresenta uma visão restrita do fazer docente, limitada pelo uso do livro didático, contrastando visivelmente com a necessidade de sua contextualização e complementação.

REFERÊNCIAS

GÓMEZ, A. P. O pensamento prático do professor – A formação do professor como profissional reflexivo. In: NÓVOA, António (org.). *Os Professores e a sua Formação*. Lisboa: Publicações Dom Quixote, 1997.

GRILLO, M. C.; GESSINGER, R. M. (Org.). Por que falar ainda em avaliação? Porto Alegre: EDIPUCRS, 2010. Disponível em: <<http://www.pucrs.br/edipucrs/porquefalaraindaemavaliacao.pdf>>. Acesso em: 11 abr. 2014

LIMA, V. M. (Org.) et al. A gestão da aula universitária na PUCRS. Porto Alegre: EDIPUCRS, 2008. Disponível em: <<http://www.pucrs.br/edipucrs/agestaodaaula.pdf>>. Acesso em: 15 abr. 2014.

QUESTÃO 33

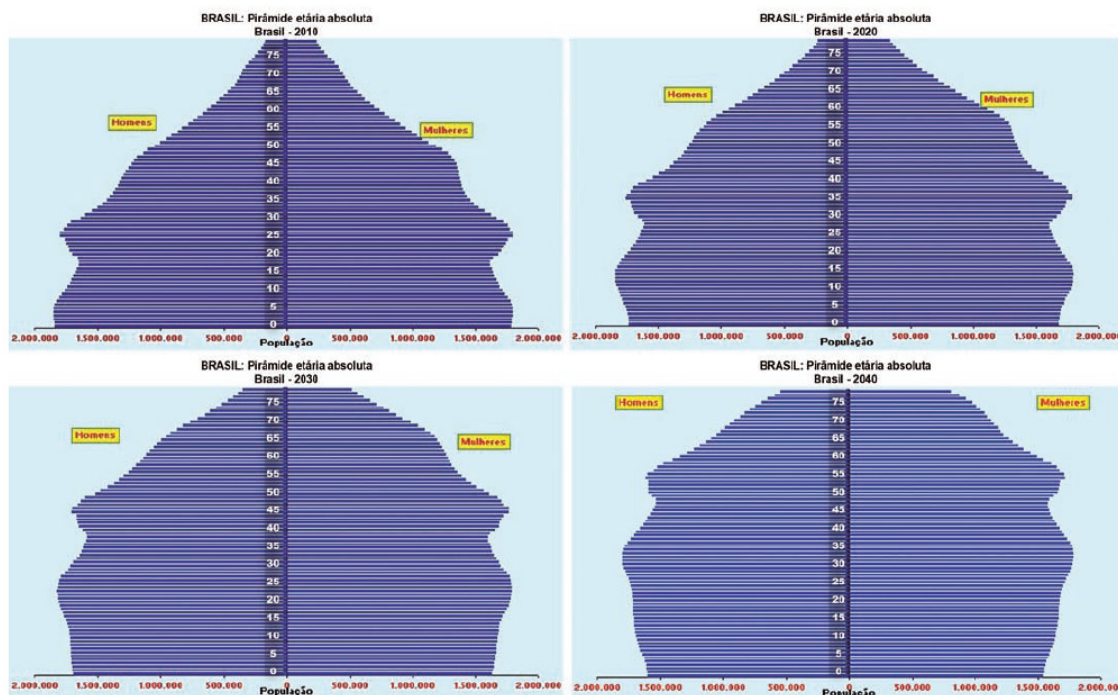


Figura. Brasil: Pirâmide Etária Absoluta (2010-2040)

Disponível em: <www.ibge.gov.br/home/estatistica/populacao/projecao_da_populacao/piramide/piramide.shtm>.
Acesso em: 23 ago. 2011.

Com base na projeção da população brasileira para o período 2010-2040 apresentada nos gráficos, avalie as seguintes asserções.

Constata-se a necessidade de construção, em larga escala, em nível nacional, de escolas especializadas na Educação de Jovens e Adultos, ao longo dos próximos 30 anos.

PORQUE

Haverá, nos próximos 30 anos, aumento populacional na faixa etária de 20 a 60 anos e decréscimo da população com idade entre 0 e 20 anos.

A respeito dessas asserções, assinale a opção correta.

- A. As duas asserções são proposições verdadeiras, e a segunda é uma justificativa correta da primeira.
- B. As duas asserções são proposições verdadeiras, mas a segunda não é uma justificativa da primeira.
- C. A primeira asserção é uma proposição verdadeira, e a segunda, uma proposição falsa.
- D. A primeira asserção é uma proposição falsa, e a segunda, uma proposição verdadeira.
- E. Tanto a primeira quanto a segunda asserções são proposições falsas.

Resposta: alternativa (D)

Autora: Rosana Maria Gessinger

COMENTÁRIO

A primeira asserção é falsa, pois, embora a projeção da população brasileira para o período de 2010-2040, apresentada no gráfico, indique aumento populacional na faixa etária de 20 a 60 anos, não é possível supor que essa faixa da população não estará escolarizada e que necessitará de escolas especializadas em Educação de Jovens e Adultos.

A segunda asserção é verdadeira, pois, comparando as regiões dos gráficos correspondentes à projeção da população de 0 a 20 anos em 2010 e 2040, percebe-se que há um decréscimo. Com relação à projeção da população de 20 a 60 anos, percebe-se que há um aumento no mesmo período.

QUESTÃO 34

Na escola em que João é professor, existe um laboratório de informática, que é utilizado para os estudantes trabalharem conteúdos em diferentes disciplinas. Considere que João quer utilizar o laboratório para favorecer o processo ensino-aprendizagem, fazendo uso da abordagem da Pedagogia de Projetos. Nesse caso, seu planejamento deve

- A. ter como eixo temático uma problemática significativa para os estudantes, considerando as possibilidades tecnológicas existentes no laboratório.
- B. relacionar os conteúdos previamente instituídos no início do período letivo e os que estão no banco de dados disponível nos computadores do laboratório de informática.
- C. definir os conteúdos a serem trabalhados, utilizando a relação dos temas instituídos no Projeto Pedagógico da escola e o banco de dados disponível nos computadores do laboratório.
- D. listar os conteúdos que deverão ser ministrados durante o semestre, considerando a sequência apresentada no livro didático e os programas disponíveis nos computadores do laboratório.
- E. propor o estudo dos projetos que foram desenvolvidos pelo governo quanto ao uso de laboratórios de informática, relacionando o que consta no livro didático com as tecnologias existentes no laboratório.

Resposta: alternativa (A)

Autora: Rosana Maria Gessinger

COMENTÁRIO

A alternativa **A** está correta, pois um dos fundamentos da Pedagogia de Projetos é a aprendizagem significativa. Além disso, a organização de um projeto pode seguir um eixo, em que o ponto de partida para a definição do trabalho a ser desenvolvido pelos alunos é a escolha do tema, a partir dos interesses e das propostas dos estudantes (HERNÁNDEZ, 1998). Nessa perspectiva, as possibilidades tecnológicas existentes no laboratório devem ser levadas em consideração, pois poderão contribuir para a execução do projeto.

As alternativas **B**, **C** e **D** estão incorretas, pois a abordagem da Pedagogia de Projetos não é compatível com uma organização prévia de conteúdos, uma vez que esses são definidos a partir da definição do tema.

A alternativa **E** está incorreta, pois não apresenta argumentos teóricos pertinentes à Pedagogia de Projetos.

REFERÊNCIA

HERNANDÉZ, F.; VENTURA, M. *A organização do currículo por projetos de trabalho: o conhecimento é um caleidoscópio*. 5. ed. Porto Alegre: Artmed, 1998.

QUESTÃO 35



QUINO. Toda a Mafalda. Trad. Andréa Stahel M. da Silva et al. São Paulo: Martins Fontes, 1993, p. 71.

Muitas vezes, os próprios educadores, por incrível que pareça, também vítimas de uma formação alienante, não sabem o porquê daquilo que dão, não sabem o significado daquilo que ensinam e quando interrogados dão respostas evasivas: “é pré-requisito para as séries seguintes”, “cai no vestibular”, “hoje você não entende, mas daqui a dez anos vai entender”. Muitos alunos acabam acreditando que aquilo que se aprende na escola não é para entender mesmo, que só entenderão quando forem adultos, ou seja, acabam se conformando com o ensino desprovido de sentido.

VASCONCELLOS, C. S. *Construção do conhecimento em sala de aula*. 13. ed. São Paulo: Libertad, 2002, p. 27-8.

Correlacionando a tirinha de Mafalda e o texto de Vasconcellos, avalie as afirmações a seguir.

- I. O processo de conhecimento deve ser refletido e encaminhado a partir da perspectiva de uma prática social.
- II. Saber qual conhecimento deve ser ensinado nas escolas continua sendo uma questão nuclear para o processo pedagógico.
- III. O processo de conhecimento deve possibilitar compreender, usufruir e transformar a realidade.
- IV. A escola deve ensinar os conteúdos previstos na matriz curricular, mesmo que sejam desprovidos de significado e sentido para professores e alunos.
- V. Os projetos curriculares devem desconsiderar a influência do currículo oculto que ocorre na escola com caráter informal e sem planejamento.

É correto apenas o que se afirma em

- A. I e III.
- B. I e IV.
- C. II e IV.
- D. I, II e III.
- E. II, III e IV.

Resposta: alternativa (D)

Autora: Ana Lúcia Souza de Freitas

COMENTÁRIO

A afirmação **I** está correta, pois o processo de conhecimento, compreendido como prática social, requer que o ensino seja exercido como ação mediadora, levando em conta os saberes que os estudantes já possuem e tendo em vista uma leitura crítica da realidade (RIOS, 2008).

A afirmação **II** está correta, pois a referência à responsabilidade docente em relação à seleção de conteúdos traz implícita a não neutralidade do conhecimento, bem como a natureza política do trabalho de ensinar, ou seja, exige uma tomada de posição do educador acerca do que ensina, por que ensina, como ensina e para que ensina (FREIRE, 1993).

A afirmação **III** está correta, pois apresenta implicitamente a compreensão sobre o processo de conhecimento como prática social, ou seja, de que as práticas pedagógicas têm uma finalidade que transcende o espaço da sala de aula (FRANCO, 2012).

As afirmações **IV** e **V** estão incorretas, visto que a formulação das mesmas contraria, de modo evidente, as afirmações anteriores.

REFERÊNCIAS

FRANCO, Maria Amélia do Rosário Santoro. *Pedagogia e Prática Docente*. São Paulo: Cortez, 2012. (Coleção Docência em Formação: Saberes Pedagógicos / Coordenação Selma Garrido Pimenta)

FREIRE, Paulo. *Professora, sim; tia, não: cartas a quem ousa ensinar*. São Paulo: Olho D'Água, 1993.

RIOS, Terezinha. *Compreender e ensinar: por uma docência da melhor qualidade*. São Paulo: Cortez, 2001.

QUESTÃO 36

O problema **P versus NP** é um problema ainda não resolvido e um dos mais estudados em Computação. Em linhas gerais, deseja-se saber se todo problema cuja solução pode ser eficientemente verificada por um computador, também pode ser eficientemente obtida por um computador. Por “eficientemente” ou “eficiente” significa “em tempo polinomial”.

A classe dos problemas cujas soluções podem ser eficientemente obtidas por um computador é chamada de **classe P**. Os algoritmos que solucionam os problemas dessa classe têm complexidade de pior caso polinomial no tamanho das suas entradas.

Para alguns problemas computacionais, não se conhece solução eficiente, isto é, não se conhece algoritmo eficiente para resolvê-los. No entanto, se para uma dada solução de um problema é possível verificá-la eficientemente, então o problema é dito estar em NP. Dessa forma, a classe de problemas para os quais suas soluções podem ser eficientemente verificadas é chamada de **classe NP**.

Um problema é dito ser **NP-completo** se pertence à classe NP e, além disso, se qualquer outro problema na classe NP pode ser eficientemente transformado nesse problema. Essa transformação eficiente envolve as entradas e saídas dos problemas.

Considerando as noções de complexidade computacional apresentadas acima, analise as afirmações que se seguem.

- I. Existem problemas na classe P que não estão na classe NP.
- II. Se o problema A pode ser eficientemente transformado no problema B e B está na classe P, então A está na classe P.
- III. Se $P = NP$, então um problema NP-completo pode ser solucionado eficientemente.
- IV. Se P é diferente de NP, então existem problemas na classe P que são NP-completos.

É correto apenas o que se afirma em

- A. I.
- B. IV.
- C. I e III.
- D. II e III.
- E. II e IV.

Gabarito: Alternativa (D)

Autor: Alfio Ricardo de Brito Martini

COMENTÁRIO

Como o enunciado da questão não traz informações suficientes para uma argumentação sólida sobre cada uma das alternativas de resposta, algumas definições auxiliares sobre as classes P e NP serão colocadas. Para maiores detalhes, sugere-se as referências [1,2,3].

Seja \mathbb{N} o conjunto dos números naturais. O tempo de execução de uma máquina de Turing M (ou qualquer outro modelo de computação) pode ser visto como uma função $f: \mathbb{N} \rightarrow \mathbb{N}$, onde $f(n)$ é o número máximo de instruções (considerando a análise do pior caso) que M usa sobre qualquer entrada de comprimento n . Além disso, com frequência estamos interessados no fato de que o tempo de execução de f possa estar *delimitado* por alguma outra função g , possivelmente mais simples de expressar. Por exemplo, considere as funções

$$f(n) = 2n^3 + 2n + 5 \quad \text{e} \quad g(n) = 10n^2 + 3.$$

Para $n \leq 4$ temos que $f(n) \leq g(n)$. Entretanto, a medida que n cresce, f cresce muito mais rapidamente do que g . O que define a taxa de crescimento de f é o seu termo mais alto. Dessa forma, dizemos que a função f é *limitada* ou *delimitada* pela função g no que se refere a sua taxa de crescimento.

Após essa breve revisão, podemos precisar melhor a classe P .

Definição 1. Dizemos que uma máquina de Turing está **polinomialmente limitada** se existe um polinômio $p(x)$ tal que, para qualquer número natural n , $t_M(n) \leq p(n)$.

Definição 2. Uma linguagem é chamada **polinomialmente decidível** se existe uma máquina de Turing determinística polinomialmente delimitada que a decide. A classe de todas as linguagens polinomialmente decidíveis é denotada por P .

A definição acima é independente do modelo formal utilizado. Em outras palavras, todos os modelos formais do mesmo algoritmo estão *polinomialmente relacionados uns com os outros* tanto quanto a complexidade computacional esteja em questão. Quando traduzimos um algoritmo para um formalismo diferente, seu tempo de execução pode aumentar, mas a taxa de crescimento está limitada por um polinômio. Dessa forma, podemos considerar que P é invariante para todos os modelos de computação. Além disso, P corresponde aproximadamente à classe de problemas que são realisticamente solucionáveis em um computador.

Exemplo 3. Existe uma classe de problemas muito interessantes e úteis, para os quais não foram encontrados até hoje algoritmos de decisão em tempo polinomial. Entretanto, eles podem ser verificados em tempo polinomial. Esta classe inclui problemas fundamentais como o problema do caixeiro viajante, da satisfatibilidade de fórmulas booleanas (SAT) e do ciclo hamiltoniano em grafos [1,2,3]. Esta classe será especificada precisamente na definição 5.

Considere agora como uma máquina de Turing não-determinística M decide uma linguagem L . Para qualquer $W \notin L$, todas as computações de M sobre W devem rejeitá-la. Para qualquer $W \in L$, pelo menos uma computação de M deve aceitá-la.

Definição 4. Dizemos que uma máquina de Turing não-determinística M é **polinomialmente limitada** se existe um polinômio $p(x)$ tal que, para qualquer string de entrada w , **pelo menos uma computação** de M na entrada w para em $p(|w|)$ passos.

De forma mais geral, abstraindo o modelo formal, pode-se conceber um algoritmo não-determinístico que, dado qualquer string $w \in L$, opera em duas fases:

1. *Adivinhar* ou *supor* um string para ser testado;
2. *Verificar* que o string satisfaz as condições do problema (se a adivinhação estiver errada, não irá satisfazer as condições).

A primeira parte do algoritmo é claramente não-determinística. Entretanto, a fase de verificação para os problemas citados acima pode ser executada em tempo polinomial por um algoritmo determinístico.

Como a discussão acima sugere, um algoritmo de verificação v toma dois argumentos, w e c . O string c é chamado de *certificado* ou *prova*. A linguagem L é dita como verificada por um algoritmo de verificação v se

$$L = \{w \mid \exists c. V(w, c) = 1\}$$

Observe que na definição de L acima, o não-determinismo está capturado pelo quantificador existencial. Agora podemos definir de forma precisa a classe NP .

Definição 5. Uma linguagem L pertence à classe NP se existe um algoritmo determinístico de tempo polinomial q tal que

$$L = \{w \mid \exists c. |c| \leq q(|w|) \wedge V(w, c) = 1\}$$

Em outras palavras, V verifica L em tempo polinomial. NP significa polinomial não-determinístico. Esse não-determinismo, como colocamos acima, vem da adivinhação do certificado c .

Definição 6. Uma função $\Sigma^* \rightarrow \Sigma^*$ é chamada **computável em tempo polinomial** se existe uma máquina de Turing determinística limitada polinomialmente que computa essa função.

Definição 7. Sejam $L, R \subseteq \Sigma^*$ duas linguagens. Dizemos que a linguagem L é **redutível em tempo polinomial para R** se existe uma função $r : \Sigma^* \rightarrow \Sigma^*$ tal que, para qualquer $w \in \Sigma$, $w \in L$ se e somente se $r(w) \in R$. A função r é chamada de **redução em tempo polinomial**.

Definição 8. Uma linguagem (problema) L é chamada de NP – **completa** se $L \in NP$ e qualquer $L' \in NP$ é redutível em tempo polinomial para L .

Seguimos agora então para as opções das questões. Os comentários serão feitos imediatamente após cada alternativa.

I. Existem problemas na classe P que não estão na classe NP .

Esta asserção é falsa. Qualquer algoritmo de tempo polinomial A que decide uma linguagem $L \in P$ pode ser facilmente convertido em um algoritmo de verificação em tempo polinomial A' que apenas ignora o segundo argumento (certificado) e simula A . Portanto, temos que $P \subseteq NP$.

II. Se o problema A pode ser eficientemente transformado no problema B e B está na classe P , então A está na classe P .

Esta asserção é verdadeira. Apenas interprete problema como decisão de uma linguagem. Logo, a afirmação de que a linguagem pode ser transformada eficientemente na linguagem B é equivalente a dizer que A é redutível em tempo polinomial para B . Agora, de acordo com a definição 7, considere o seguinte algoritmo: suponha que temos um algoritmo A_B que decide B em tempo polinomial. Então o seguinte algoritmo A_A decide A também em tempo polinomial: dada qualquer entrada $w \in A$, computar $r(w)$ em tempo polinomial. Então chamar o algoritmo polinomial de B para determinar se $r(w) \in B$. Se $r(w) \in B$, então $w \in A$. Do contrário $w \notin A$. Isso significa que A , através de B , é polinomialmente decidível e, portanto está em P .

III. Se $P = NP$, então um problema NP -completo pode ser solucionado eficientemente.

Esta é uma asserção condicional. Não se sabe até hoje se o antecedente é verdadeiro ou falso. (embora a probabilidade de que seja falsa é grande). Se ela for falsa, então a implicação é trivialmente verdadeira. Se ela for verdadeira, então por definição, qualquer problema em P e, portanto em NP pode ser solucionado polinomialmente, isto é, de forma eficiente. Desta forma, como qualquer problema NP -completo está em NP , segue que ele, por estar também em P , teria também uma solução eficiente. Logo, a asserção é verdadeira.

IV. Se $P \neq NP$, então existem problemas na classe P que são NP -completos.

Esta é também uma asserção condicional. Provamos que ela é falsa, mostrando que ao assumirmos o antecedente e o consequente, derivamos uma contradição. Da hipótese de que $P \neq NP$ e levando em conta que $P \subseteq NP$ (ver alternativa I), segue que $P \subset NP$. Desta forma, seja L_p um problema em P que seja NP -completo. Pela definição 8 todo problema em NP pode ser reduzido de forma eficiente a L_p . Daí segue, pela alternativa II, que todo problema em NP estaria em P , isto é, que $P = NP$. Mas isto é uma contradição com a hipótese de que $P \neq NP$. Portanto, esta afirmação é falsa.

Da discussão acima segue que apenas as afirmativas II e III são verdadeiras.

REFERÊNCIAS

- KINBER, E.; SMITH, C. *Theory of Computing: A Gentle Introduction*. Upper Saddle River: Prentice-Hall, 2001.
- LEWIS, H.; PAPADIMITRIOU, C. *Elementos de Teoria da Computação*. Porto Alegre: Bookman, 2000.
- SIPSER, M. *Introdução à Teoria da Computação*. São Paulo: Thomson Learning, 2007.

QUESTÃO 37

Escopo dinâmico: para as linguagens com escopo dinâmico, a vinculação das variáveis ao escopo é realizada em tempo de execução. (...) Se uma variável é local ao bloco, então o uso da dada variável no bloco será sempre vinculado àquela local. Contudo, se a variável for não local, a sua vinculação depende da ordem de execução, a última vinculada na execução. A consequência disso é que, em um mesmo bloco de comandos, um identificador pode ter significados diferentes, e o programador precisa ter a ideia precisa de qual variável está sendo usada.

MELO, A. C. V.; SILVA, F. S. C. *Princípios de Linguagens de Programação*. São Paulo: Edgard Blücher, 2003. p. 65.

Suponha que uma linguagem de programação tenha sido projetada com vinculação e verificação estáticas para tipos de variáveis, além de passagem de parâmetros por valor.

Também é exigido pela especificação da linguagem que programas sejam compilados integralmente e que não é permitido compilar bibliotecas separadamente. Durante uma revisão da especificação da linguagem, alguém propôs que seja adicionado um mecanismo para suporte a variáveis com escopo dinâmico.

A respeito da proposta de modificação da linguagem, analise as seguintes afirmações.

- I. As variáveis com escopo dinâmico podem ser tratadas como se fossem parâmetros para os subprogramas que as utilizam, sem que o programador tenha que especificá-las ou declarar seu tipo (o compilador fará isso). Assim, eliminasse a necessidade de polimorfismo e é possível verificar tipos em tempo de compilação.
- II. Como diferentes subprogramas podem declarar variáveis com o mesmo nome mas com tipos diferentes, se as variáveis com escopo dinâmico não forem declaradas no escopo onde são referenciadas, será necessário que a linguagem suporte polimorfismo de tipos.
- III. Se as variáveis dinâmicas forem declaradas tanto nos escopos onde são criadas como nos subprogramas em que são referenciadas, marcadas como tendo escopo dinâmico, será possível identificar todos os erros de tipo em tempo de compilação.

É correto apenas o que se afirma em:

- A. I.
- B. II.
- C. I e III.
- D. II e III.
- E. I, II e III.

Resposta: alternativa (D)

Autor: Marco Gonzalez

COMENTÁRIO

Conforme Ghezzi e Jazayeri (GHEZZI; JAZAYERI, 1998, p. 93-96), linguagens como APL, SNOBOL4 e LISP adotam recursos para variáveis com escopo dinâmico. Uma propriedade dinâmica, em geral, implica amarração em tempo de execução, inviabilizando-a em tempo de compilação. Essa implicação é válida também no caso de variáveis com escopo dinâmico.

Para exemplificar, vamos considerar o trecho de programa a seguir, exemplificado em Ghezzi e Jazayeri (GHEZZI; JAZAYERI, 1998, p. 95), com sintaxe da linguagem C e semântica da linguagem APL:

```
sub2() {  
    declare x;  
    ...  
    ... x ...;  
    ... y ,,,;  
    ...  
}  
sub1() {  
    declare y;  
    ...  
    ... x ...;  
    ... z ...;  
    sub2();  
    ...  
}  
main() {  
    declare x, y, z;  
    z = 0;  
    x = 5;  
    y = 7;  
    sub1;  
    sub2;  
    ...  
}
```

Nesse exemplo, *y* é variável local à rotina *sub1*, *x* é variável local à rotina *sub2*, e *x*, *y* e *z* são variáveis locais à rotina principal, a *main*. Nesse contexto, qualquer acesso a uma variável em determinado ponto do programa, que não esteja declarada de forma local na rotina onde esse ponto está inserido, esse acesso é implicitamente assumido como referente a uma variável não local. No caso do escopo dinâmico, sabe-se que esse escopo depende da ordem de execução das unidades de programação e não da estrutura estática do programa.

No exemplo apresentado, quando *sub1* é ativada a partir da *main*, as referências a *x* e a *z* são não locais, ou seja, são globais atendendo à declaração dessas variáveis na *main*. Quando *sub2* é ativada a

partir de *sub1*, a referência não local a *y*, em *sub2*, é amarrada à sua mais recente declaração, ou seja, ao objeto de dados associado a *y* local em *sub1*. Quando *sub2* é ativada a partir da *main*, o acesso a *y*, em *sub2*, fará referência, agora, ao objeto de dado associado a *y* local na *main*.

A afirmação I não é verdadeira porque, ao adotar variáveis com escopo dinâmico, o escopo de uma variável fica dependente das ativações ou não das rotinas em questão, ou seja, é definido em função da execução do programa. Assim, o compilador fica impossibilitado da verificação correta de tipo, que só poderá ser realizada em tempo de execução, conforme Ghezzi e Jazayeri (GHEZZI, 1998, p. 94).

Com o exemplo apresentado, fica claro que a afirmação II está correta, pois o acesso correto a variáveis com o mesmo nome, como é o caso de *y*, mas declaradas em rotinas diferentes, é viabilizado através do recurso de escopo dinâmico, pois permitirá referências a declarações dessas variáveis nas respectivas rotinas.

No caso da afirmação III, no contexto proposto há restrição ao aspecto dinâmico do escopo. Ressaltamos esse fato pois ele se constitui em condição e não em conclusão na afirmação. Assim, a afirmação III está correta porque a verificação de erros de tipo poderá ser realizada em tempo de compilação se uma variável, além da declaração existente no escopo em que é criada, possuir declaração também na rotina em que é acessada (que não é o caso do exemplo apresentado aqui). Dessa forma, a estrutura estática do programa permitiria ao compilador associar corretamente a variável, no seu acesso, às propriedades do correto objeto de dado em uso. No caso do exemplo apresentado, a verificação em tempo de compilação se viabilizaria ao ser incluída em *sub2* uma declaração para *y*, especificando o caráter do escopo dinâmico e indicando quais propriedades estariam sendo adotadas ali para *y* de forma que o compilador pudesse verificá-las.

REFERÊNCIA

GHEZZI, C.; JAZAYERI, M. *Programming language concepts*. 3. ed. Nova York: John Wiley, 1998.

QUESTÃO 38

É comum que linguagens de programação permitam a descrição textual de constantes em hexadecimal, além de descrições na base dez. O compilador para uma linguagem que suporte constantes inteiras em hexadecimal precisa diferenciar inteiros em base dez dos números hexadecimais que não usam os dígitos de A a F. Por exemplo, a sequência de caracteres 12 pode ser interpretada como doze em base dez ou como dezoito em hexadecimal. Uma maneira de resolver esse problema é exigindo que as constantes em hexadecimal terminem com o caracter “x”. Assim, não há ambiguidade, por exemplo, no tratamento das sequências 12 e 12x.

A gramática a seguir descreve números inteiros, possivelmente com o símbolo “x” após os dígitos. Os não terminais são M , N , E e os terminais são x e d , em que d representa um dígito.

$$M \rightarrow E$$
$$M \rightarrow N$$
$$E \rightarrow Nx$$
$$N \rightarrow Nd$$
$$N \rightarrow d$$

Durante a construção de um autômato LR para essa gramática, os seguintes estados são definidos:

$$e_0:$$
$$M' \rightarrow \cdot M$$
$$M \rightarrow \cdot E$$
$$M \rightarrow \cdot N$$
$$E \rightarrow \cdot Nx$$
$$N \rightarrow \cdot Nd$$
$$N \rightarrow \cdot d$$
$$e_1(e_0 N):$$
$$M \rightarrow N \cdot$$
$$M \rightarrow N \cdot x$$
$$M \rightarrow N \cdot d$$

A respeito dessa gramática, analise as seguintes asserções e a relação proposta entre elas.

A gramática descrita é do tipo LR(0).

PORQUE

É possível construir um autômato LR(0), determinístico, cujos estados incluem e_0 e e_1 acima descritos.

Acerca dessas asserções, assinale a opção correta.

- A. As duas asserções são proposições verdadeiras, e a segunda é uma justificativa correta da primeira.
- B. As duas asserções são proposições verdadeiras, mas a segunda não é uma justificativa correta da primeira.
- C. A primeira asserção é uma proposição verdadeira, e a segunda, uma proposição falsa.
- D. A primeira asserção é uma proposição falsa, e a segunda, uma proposição verdadeira.
- E. Tanto a primeira quanto a segunda asserções são proposições falsas.

Resposta: alternativa (E)

Autor: Alexandre Agustini

COMENTÁRIO

O *Relatório Síntese 2011: Computação* apresenta o seguinte comentário:

[...] A questão de número 38 foi a mais difícil dentre as 26 questões específicas válidas, com baixo índice de facilidade, apenas 8,0% de acertos. Essa questão apresentou poder discriminatório igualmente baixo, 0,15, o que comprova ter sido esta a mais difícil para os estudantes (p. 63).

Dessa forma, foi considerada muito difícil e, por não possuir índice de discriminação, foi eliminada do cálculo da nota final do Enade 2011.

A Questão 38 é do tipo asserção-razão. Nesse modelo de questão, são apresentadas duas afirmativas (asserções) que podem ser proposições verdadeiras ou falsas, assim como podem ou não estabelecer relações entre si. Dessa forma, o primeiro objetivo é determinar se as afirmativas apresentadas são verdadeiras ou falsas. No caso de serem as duas proposições verdadeiras, estabelecer se a segunda justifica ou não a primeira.

A primeira asserção é “A gramática descrita é LR(0)”. Uma gramática G é dita uma gramática LR(0) se as seguintes propriedades forem válidas:

1. o símbolo inicial de G não aparece no lado direito de uma produção;
2. todo prefixo γ de G onde $A \rightarrow \alpha$ é um item completo e único, ou seja, não há outros itens completos (e não existem itens com um terminal à direita do ponto) que são válidos para γ .

A segunda propriedade pode ser resumida como: não há estado de redução que também contém ação para empilhar e em cada estado de redução só é possível reduzir de acordo com uma única produção. Com isso, fica claro que a gramática apresentada não é LR(0), pois a regra falha no estado e_i do autômato.

A segunda proposição é “É possível construir um autômato LR(0), determinístico, cujos estados incluem e_0 e e_i acima”. Segundo a definição de autômato determinístico apresentado por Aho (2008), o autômato gerado não será determinista, pois não há “estados mortos”, logo, haverá pares “estado-palavra” para os quais não há estado seguinte. Portanto, as duas afirmações são falsas e a resposta correta é a alternativa **E**.

REFERÊNCIAS

AHO, A. V.; SETHI, S.; ULLMAN, J. D. *Compiladores: princípios, técnicas e ferramentas*. 2. ed. São Paulo: Pearson, 2008. 634p.

HOPCROFT, John E.; MOTWANI, Rajeev; ULLMAN, Jeffrey D. *Introdução à teoria de autômatos, linguagens e computação*. Rio de Janeiro: Elsevier, 2003. 560 p.

INEP. Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira, Ministério da Educação, Brasil. *Relatório Síntese 2011: Computação 2013*. Disponível em: <<http://portal.inep.gov.br/enade/relatorio-sintese-2011>>. Acesso em: abr. 2014.

QUESTÃO 39

O conceito de Tipo de Dados Abstrato (TDA) é popular em linguagens de programação. Nesse contexto, analise as afirmativas a seguir.

- I. A especificação de um TDA é composta das operações aplicáveis a ele, da sua representação interna, e das implementações das operações.
- II. Dois mecanismos utilizáveis na implementação de um TDA em programas orientados a objetos são a composição e a herança.
- III. Se S é um subtipo de outro T, então entidades do tipo S em um programa podem ser substituídas por entidades do tipo T, sem alterar a corretude desse programa.
- IV. O encapsulamento em linguagens de programação orientadas a objetos é um efeito positivo do uso de TDA.

É correto apenas o que se afirma em

- A. I.
- B. II.
- C. I e III.
- D. II e IV.
- E. III e IV.

Resposta: alternativa (D)

Autor: Júlio Henrique Araújo Pereira Machado

COMENTÁRIO

A questão envolve conceitos básicos associados a algoritmos e especificação e tipos de dados, bem como a implementação dos mesmos em linguagens de programação. Mais especificamente, o tópico abordado é Tipos de Dados Abstratos (TDA) e sua implementação em linguagens do paradigma de Orientação a Objetos.

A construção da resposta correta consiste em identificar cada uma das afirmativas apresentadas como sendo verdadeira ou falsa.

A afirmativa I deve ser considerada falsa, uma vez que a definição para TDA apresentada traz uma informação incorreta, qual seja, a de que um tipo abstrato deve conter a sua representação interna dos dados. Ora, tal afirmação vai contra a própria definição de “abstrato”, que traz consigo o conceito de que um TDA é independente de sua implementação concreta, ou seja, que o TDA pode vir a ser implementado de diversas formas alternativas (por exemplo, no caso do tipo Lista, que pode ser implementado via estruturas encadeadas ou estruturas estáticas de *array*). Como exemplo concreto, a Figura 1 traz a associação entre a interface *List<E>* e as possíveis implementações *LinkedList<E>* e *ArrayList<E>* da linguagem Java.

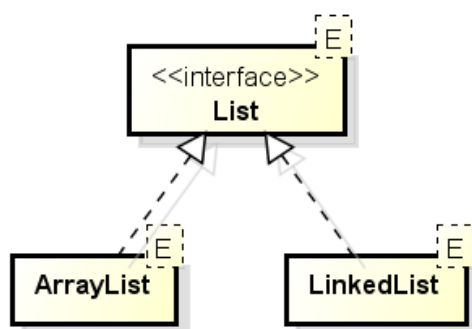


Figura 1. Implementação de TDA Lista em Java.

A afirmativa II é correta, pois os mecanismos da Orientação a Objetos citados, composição e herança, são básicos na construção de classes. Para exemplificar o mecanismo de composição, tem-se o exemplo de implementação de uma Pilha a partir de uma Lista. Já o mecanismo da herança pode ser exemplificado na implementação de uma Lista de Tamanho Limitado a partir de uma Lista. Como exemplo concreto, observe a implementação parcial em Java a seguir:

```

public class Lista<T> {
    ...
    public void inserir(T elemento) {...}
    public int tamanho() {...}
    ...
}

public class Pilha<T> {
    private Lista<T> elementos;
    ...
    public void empilhar(T elemento) {
        elementos.inserir(elemento);
    }
    public int tamanho() {
        return elementos.tamanho();
    }
}

public class ListaTamanhoLimitado<T> extends Lista<T> {
    ...
    public int tamanhoMaximo() {...}
    public void inserir(T elemento) throws Exception{
        if(tamanho() < tamanhoMaximo())
            super.inserir(elemento);
        else
            throw new Exception();
    }
    ...
}

```

A afirmativa III é falsa, pois apresenta o conceito de polimorfismo na herança de maneira incorreta. Dado S, um subtipo de T, o correto é afirmar que qualquer ocorrência de T pode ser substituída por uma ocorrência de S. Ou seja, todo subtipo S é também um tipo compatível com o supertipo T. Como exemplo concreto, observe a implementação parcial em Java a seguir:

```

public class T {...}

public class S extends T {...}

public class Exemplo {
    public static void main(String[] args) {
        T ref1 = new T(); // correto
        T ref2 = new S(); // correto
        S ref3 = new S(); // correto
        S ref4 = new T(); // incorreto
    }
}

```

A afirmativa IV deve ser considerada como verdadeira, já que a afirmativa está diretamente relacionada à definição de Tipo de Dados Abstrato. Quando um TDA é implementado de acordo com o paradigma de Orientação a Objetos, sua representação interna (estrutura de dados e operações que não dizem respeito à interface pública do objeto) deve permanecer “escondida” do usuário do objeto, ou seja, não pertence à interface pública e, portanto, está encapsulada. Além disso, o correto encapsulamento permite que a estrutura interna de implementação mude completamente sem afetar a interface pública do objeto em questão. Um exemplo concreto na linguagem Java é o mesmo apresentado na Figura 1.

REFERÊNCIAS

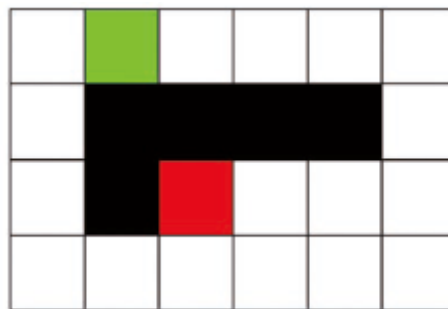
CORMEN, Thomas H. *et al. Introduction to algorithms*. Cambridge: The MIT Press, 2009.

FORBELLONE, André L. V; EBERSPÄCHER, Henri F. *Lógica de programação: a construção de algoritmos e estruturas de dados*. São Paulo: Pearson, 2011.

GOODRICH, Michael T; TAMASSIA, Roberto. *Estruturas de dados e algoritmos em Java*. Porto Alegre: Bookman, 2013.

QUESTÃO 40

Considere que a figura abaixo corresponde ao cenário de um jogo de computador. Esse cenário é dividido em 24 quadrados e a movimentação de um personagem entre cada quadrado tem custo 1, sendo permitida apenas na horizontal ou na vertical. Os quadrados marcados em preto correspondem a regiões para as quais os personagens não podem se mover.



Nesse cenário, o algoritmo A^* vai ser usado para determinar o caminho de custo mínimo pelo qual um personagem deve se mover desde o quadrado verde até o quadrado vermelho. Considere que, no A^* , o custo $f(x) = g(x) + h(x)$ de determinado nó x é computado somando-se o custo real $g(x)$ ao custo da função heurística $h(x)$ e que a função heurística utilizada é a distância de Manhattan (soma das distâncias horizontal e vertical de x até o objetivo). Desse modo, o custo $f(x)$ do quadrado verde é igual a

- A. 2.
- B. 3.
- C. 5.
- D. 7.
- E. 9.

Resposta: alternativa (B)

Autores: Bernardo Copstein e Michael da Costa Mora

COMENTÁRIO

O algoritmo A* é um algoritmo de busca que determina o caminho de menor custo de um nodo inicial até um nodo objetivo e tem seu caminho de busca guiado por uma heurística que estima a distância do estado corrente até o destino. À medida que o algoritmo percorre as possibilidades, opta sempre pelo caminho que apresenta o menor custo. O custo de cada nodo x é fornecido por uma função de custo (normalmente denotada por $f(x)$) que é usada para determinar a ordem em que a pesquisa visita os nodos da árvore. A função de custo corresponde à soma de duas funções:

- a função de custo passado, que corresponde a distância do nodo de origem até o nodo atual (normalmente denotada por $g(x)$);
- a função de custo futuro, que corresponde a uma estimativa heurística admissível da distância que falta percorrer até o objetivo (normalmente denotada por $h(x)$).

Para aplicações de determinação de caminho, $h(x)$ pode ser equivalente, por exemplo, a distância em linha reta até o objetivo, visto que uma linha reta é a menor distância possível entre dois pontos.

A questão pede que se calcule o custo $f(x)$ da posição verde, tendo como destino a posição vermelha (ver Figura 1), no contexto do algoritmo A*. Coloca que, para o cálculo de $h(x)$, usa-se a distância de Manhattan, ou seja, consideram-se apenas deslocamentos na vertical ou horizontal no cálculo da distância.

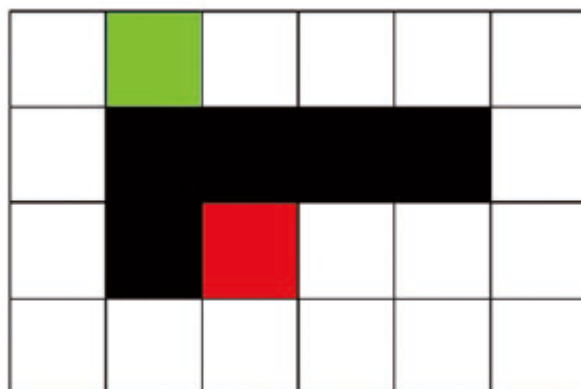


Figura 1. Problema proposto.

No caso a posição verde é a posição de partida, logo, para essa posição, o custo acumulado para se chegar até a mesma é zero, ou seja, $g(x) = 0$. Já o custo da heurística – estimativa da distância que falta percorrer – é igual a três, pois, considerando a distância de Manhattan, apenas três posições separam a posição verde da posição vermelha, ou seja, $h(x) = 3$. Sendo assim, como $f(x) = g(x) + h(x)$, o custo da posição verde é igual a $f(x) = 0 + 3$, $f(x) = 3$ (alternativa B).

O que pode causar certa confusão, em um primeiro momento, é o fato de que o problema coloca que as posições marcadas em preto correspondem a obstáculos que não podem ser transpostos. Dessa maneira, intuitivamente, calcula-se que o peso da posição verde é 7 (alternativa D), pois essa é a menor distância entre as posições verde e vermelho contornando-se os obstáculos. Ocorre que a heurística $h(x)$ trabalha com uma **estimativa** de distância, ignorando os obstáculos (se $h(x)$ pudesse antever o

menor caminho, o problema se resolveria em um único passo). A ideia do algoritmo A^* é considerar $h(x)$ de maneira a determinar o quão distante se está, em cada passo, do suposto menor caminho.

REFERÊNCIA

RUSSELL, S.; NORVIG, P. *Inteligência Artificial*. 2. ed. Rio de Janeiro: Editora Campus, 2004.

QUESTÃO 41

A figura abaixo ilustra a tentativa de se utilizar um filtro digital no domínio da frequência, para suavizar o sinal bidimensional de entrada que está no domínio do espaço.



A partir do resultado obtido no processo de filtragem, analise as seguintes asserções e a relação proposta entre elas.

O sinal de saída possui as características de um sinal processado por um filtro passa-baixa ideal.

PORQUE

Embora suavizado, o sinal de saída evidencia a presença do efeito de ringing, que é típico de um sinal convolucionado pela função sinc no domínio do espaço.

Acerca dessas asserções, assinale a opção correta.

- A. As duas asserções são proposições verdadeiras, e a segunda é uma justificativa correta da primeira.
- B. As duas asserções são proposições verdadeiras, mas a segunda não é uma justificativa correta da primeira.
- C. A primeira asserção é uma proposição verdadeira e a segunda, uma proposição falsa.
- D. A primeira asserção é uma proposição falsa e a segunda, uma proposição verdadeira.
- E. Tanto a primeira quanto a segunda asserções são proposições falsas.

Resposta: alternativa (A)

Autor: Dênis Fernandes

COMENTÁRIO

A questão aborda conceitos básicos de processamento digital de sinais (DSP) apresentando uma aplicação referente à filtragem de uma imagem bidimensional.

Filtros digitais passa-baixos são frequentemente utilizados em processamento de imagens com o objetivo de obter a suavização das mesmas. Essa suavização implica atenuação do ruído nas imagens e no borrimento (nem sempre desejado) das bordas dos objetos presentes.

A resposta em frequência $H(u,v)$ de um filtro passa-baixos bidimensional ideal é apresentada na Figura 1, onde u e v são os eixos de frequência. Na mesma figura, também é apresentado um corte transversal da resposta em frequência (radialmente simétrica nesse caso), onde se observa claramente que as amplitudes das componentes com frequências espaciais até a frequência de corte D_0 não são modificadas. Já as componentes com frequência espacial acima da frequência de corte são eliminadas, caracterizando uma filtragem passa-baixos ideal.

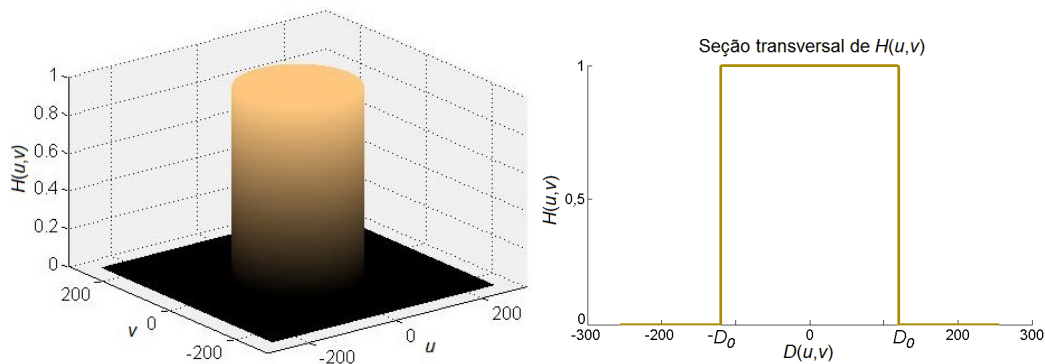


Figura 1. Resposta em frequência $H(u,v)$ de um filtro passa-baixos ideal bidimensional.

A questão apresenta uma estratégia para filtragem de uma imagem em que a mesma é transformada para o domínio da frequência utilizando técnicas de Fourier. A imagem no domínio da frequência é então multiplicada pela resposta em frequência do filtro desejado. A aplicação da transformação inversa, do domínio da frequência para o domínio do espaço, resulta uma versão filtrada da imagem original.

Das propriedades básicas das técnicas de Fourier, sabe-se que a um produto no domínio da frequência está associada uma operação de convolução no domínio do tempo ou do espaço (no caso de uma imagem). Sabe-se também que uma resposta em frequência na forma de um pulso retangular corresponde a um sinal do tipo *sinc* no domínio do tempo (unidimensional) ou do espaço (bidimensional). No caso de uma dimensão, a resposta à amostra unitária do filtro digital passa-baixos ideal com frequência de corte D_0 tem a forma apresentada na equação (1). A convolução de $h(n)$ com o sinal a ser filtrado resultará no surgimento de uma ondulação relacionada com a frequência de corte do filtro. Tal efeito é similarmente observado no caso de sinais bidimensionais, como no caso de imagens.

$$h(n) = \frac{D_0}{\pi} \text{sinc}(D_0 n) = \frac{\text{sen}(D_0 n)}{\pi n}$$

Considerando o exposto, pode-se afirmar que a imagem de saída do filtro apresentada na questão possui características que indicam que a mesma tenha sido processada por um filtro passa-baixos ideal (mas não necessariamente). Como justificativas para essa conclusão, aponta-se a suavização

da fronteira entre as regiões clara e escura da imagem e a oscilação da intensidade (nível de cinza) decorrente da convolução da imagem original com uma função do tipo *sinc* bidimensional.

REFERÊNCIAS

GONZALEZ, Rafael C. *Processamento de imagens digitais*. São Paulo: Blucher, 2007. 509 p.

HAYES, Monson H. *Teoria e problemas de processamento digital de sinais*. Porto Alegre: Bookman, 2006. 466 p.

LATHI, Bhagwandas Pannalal. *Sinais e sistemas lineares*. 2. ed. Porto Alegre: Bookman, 2008. 856 p.

QUESTÃO 42

Sabendo que a principal tarefa de um sistema será de classificação em domínios complexos, um gerente de projetos precisa decidir como vai incorporar essa capacidade em um sistema computacional a fim de torná-lo inteligente. Existem diversas técnicas de inteligência computacional / artificial que possibilitam isso.

Nesse contexto, a técnica de inteligência artificial mais indicada para o gerente é

- A. lógica nebulosa.
- B. árvores de decisão.
- C. redes neurais artificiais.
- D. ACO (do inglês, *Ant-Colony Optimization*).
- E. PSO (do inglês, *Particle Swarm Optimization*).

Resposta: alternativa (C)

Autora: Renata Vieira

COMENTÁRIO

Essa questão pode ser resolvida por eliminação e refinamento. Primeiro pode-se eliminar as técnicas que não são apropriadas para a tarefa de CLASSIFICAÇÃO, opções A, D e E. Entre as opções restantes, redes neurais tendem a ser aplicadas em domínios mais complexos (com alta dimensionalidade de dados, por exemplo, imagens), por lidarem melhor com variáveis numéricas, em especial se os dados possuem mais ruído, ou até mesmo contradição. As árvores de decisão apesar de poderem ser usadas com dados numéricos, tendem a ser mais bem aproveitadas para classificar dados que podem ser expressos de forma booleana (verdadeiro ou falso) ou simbólicos.

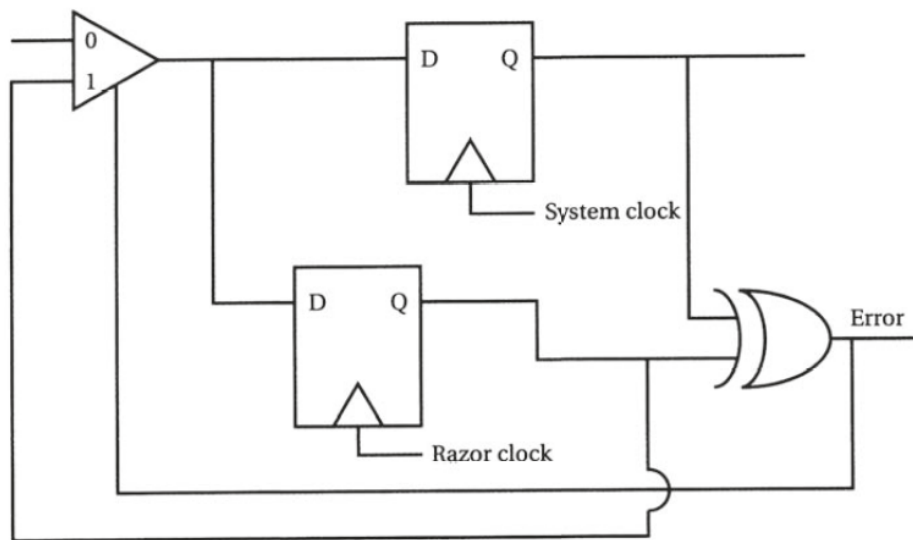
REFERÊNCIAS

RUSSEL, S.; NORVIG, P. *Artificial Intelligence: a modern approach*. Prentice-Hall, 1995. 931p.

ALPAYDIN, Ethem. *Introduction to Machine Learning*. The MIT Press, Cambridge, 2004. 415p.

QUESTÃO 43

O razor é uma arquitetura para desempenho better-than-worst-case que usa um registrador especializado, mostrado na figura, que mede e avalia os erros.



O registrador do sistema mantém o valor chaveado e é comandado por um clock de sistema better-than-worst-case. Um registrador adicional é comandado separadamente por um clock ligeiramente atrasado com relação ao do sistema. Se os resultados armazenados nos dois registradores são diferentes, então um erro ocorreu, provavelmente devido à temporização. A porta XOR detecta o erro e faz com que este valor seja substituído por aquele no registrador do sistema.

Wolf, W. High-performance embedded computing: architectures, applications, and methodologies. Morgan Kaufmann, 2007

Considerando essas informações, analise as afirmações a seguir.

- I. Sistemas digitais são tradicionalmente concebidos como sistemas assíncronos regidos por um clock.
- II. Better-than-worst-case é um estilo de projeto alternativo em que a lógica detecta e se recupera de erros, permitindo que o circuito possa operar com uma frequência maior.
- III. Nos sistemas digitais, o período de clock é determinado por uma análise cuidadosa para que os valores sejam armazenados corretamente nos registradores, com o período de clock alargado para abranger o atraso de pior caso.

É correto o que se afirma em

- A. I, apenas.
- B. III, apenas.
- C. I e II, apenas.
- D. II e III, apenas.
- E. I, II e III.

Resposta: alternativa (D)

Autor: Fernando Gehm Moraes

COMENTÁRIO

Better Than Worst-Case Design é uma técnica que separa os elementos de memória em dois componentes: um flip-flop principal e um flip-flop de verificação (*shadow*). Essa técnica permite determinar a frequência de operação de cada estágio de um circuito pelo atraso médio, e não pelo pior caso. O objetivo da técnica é a detecção de erros em caso de falha de atraso. Em situações normais (sem erros), ambos flip-flops geram a mesma saída. O flip-flop de verificação é controlado por um sinal de relógio atrasado, mas com mesma frequência (*razor clock*). Quando há um erro devido a atraso na lógica combinacional que gera dados para os flip-flops (esses erros de atraso podem ser causados, por exemplo, por redução na tensão de alimentação ou aumento da frequência de operação), haverá diferença na saída dos flip-flops, com a sinalização de erro na saída da porta XOR. Assim, os dados corretos estarão armazenados na saída do flip-flop de verificação, sendo esses direcionados para a saída do circuito através desse flip-flop.

Afirmações da questão e comentários:

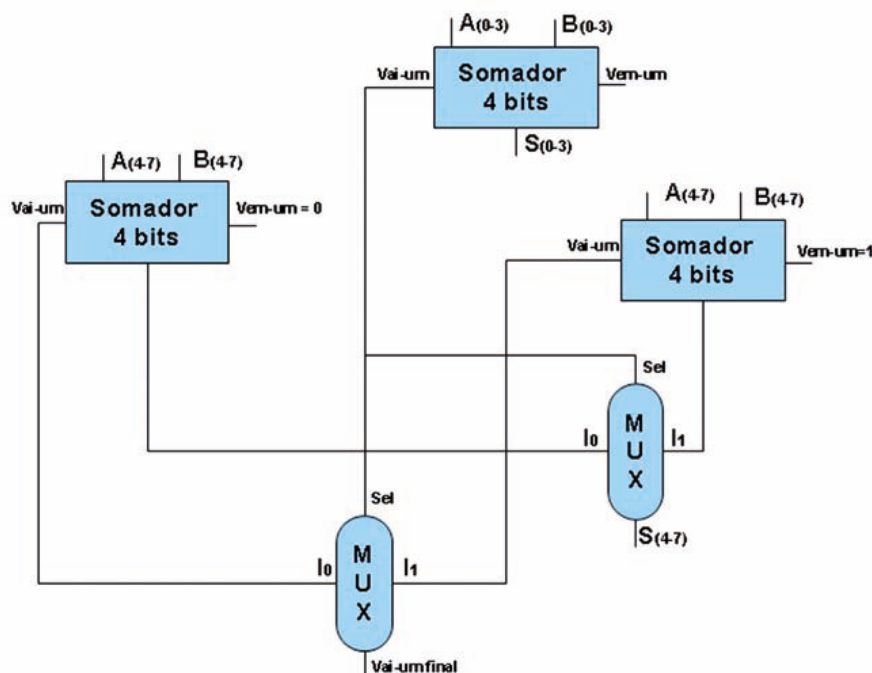
- I - FALSA. A grande maioria dos sistemas digitais é síncrona, com um ou mais sinais de relógio (*clock*). Circuitos assíncronos são menos utilizados devido à falta de ferramentas de projeto e à complexidade de projeto maior que em sistemas síncronos.
- II - VERDADEIRA. Como explicado anteriormente, o circuito pode operar no caso médio e não no pior caso, e a porta XOR indicar erro (saídas diferentes nos flip-flops) devido a um período de *clock* menor (maior frequência). Assim a resposta certa estará armazenada no flip-flop de verificação, podendo assim o circuito se recuperar de erros.
- III - VERDADEIRA. Como pode ser inferido da explicação acima, o período de *clock* deve ser determinado calculando-se o pior caso de atraso em cada estágio (chamados *corner cases* de temperatura, tensão, frequência).

REFERÊNCIA

ERNST, Dan et al. Razor: A Low-Power Pipeline Based on Circuit-Level Timing Speculation. *International Symposium on Microarchitecture (MICRO)*, 2003, p. 7-18.

QUESTÃO 44 (ANULADA)

A utilização dos somadores completos em cascata no projeto de Unidades Lógicas Aritméticas pode comprometer o seu desempenho, uma vez que o sinal de vai-um final deve propagar por todos os somadores, desde as entradas dos bits menos significativos. Esse caminho crítico insere um atraso no sistema que compromete o projeto de ULAs rápidas. Para reduzir esse atraso, mecanismos de predição de vai-um podem ser usados. Um esquema bem simples de predição de vai-um para um somador de 8 bits é apresentado na figura a seguir.



Os 4 bits mais significativos são somados de forma redundante, considerando o vem-um 0 no primeiro somador e vem-um igual a 1 no segundo somador. A saída dos somadores é selecionada a partir de um multiplexador, que é acionado pelo vai-um resultado da soma dos 4 bits menos significativos. Como os 3 somadores podem realizar as operações ao mesmo tempo, o multiplexador pode entregar o resultado mais rapidamente.

Considere as seguintes equações dos somadores:

$$S = A \oplus B \oplus VemUm \text{ e } VaiUm = A.B + VemUm.A + VemUm.B$$

Considere, ainda, a equação dos multiplexadores a seguir:

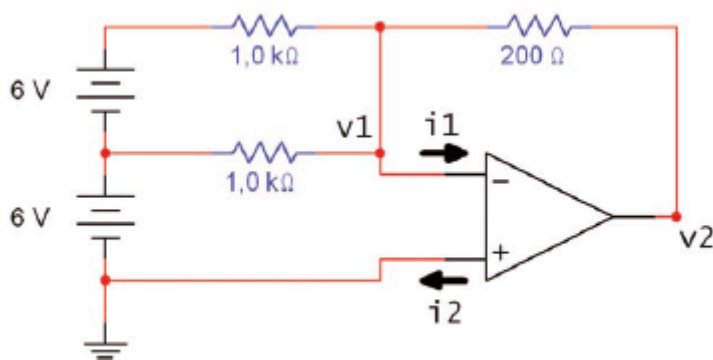
$$S = \overline{Sel}.I0 + Sel.I1$$

Suponha que o somador de 8 bits tem predição de vai-um baseada na duplicação da soma dos 4 bits mais significativos e que 7 ns é o tempo de atraso de propagação por nível de porta AND, OR e XOR. Desconsiderando os inversores, o aumento do número de portas e a redução do tempo de propagação podem ser expressos, em porcentagem, como aumento de

- A.** 26 portas, representando 65% de acréscimo no número de portas e redução de 112 ns para 70 ns, redução de 47% do tempo para a execução da soma.
- B.** 20 portas, representando 50% de acréscimo no número de portas e redução de 112 ns para 70 ns, redução de 47% do tempo para a execução da soma.
- C.** 26 portas, representando 65% de acréscimo no número de portas e redução de 112 ns para 56 ns, redução de 50% do tempo para a execução da soma.
- D.** 20 portas, representando 50% de acréscimo no número de portas e redução de 112 ns para 56 ns, redução de 50% do tempo para a execução da soma.
- E.** 26 portas, representando 65% de acréscimo no número de portas. Não há redução no tempo de atraso de propagação para a execução da soma.

QUESTÃO 45

Os amplificadores operacionais, como ilustra a figura a seguir, são componentes úteis em diversas aplicações.



Considerando que o amplificador operacional do circuito é ideal, avalie as seguintes afirmativas.

- I. A corrente i_1 é idealmente nula.
- II. A corrente i_2 é idealmente nula.
- III. O circuito exemplifica um seguidor de tensão.
- IV. A diferença de potencial entre o ponto v1 e o ponto terra do circuito é idealmente nula.
- V. A diferença de potencial entre o ponto v2 e o ponto terra do circuito é de +3,6 V.

É correto apenas o que se afirma em

- A. I, II e III.
- B. I, II e IV.
- C. I, III e V.
- D. II, IV e V.
- E. III, IV e V.

Resposta: alternativa (B)

Autor: Dênis Fernandes

COMENTÁRIO

A questão aborda conceitos básicos de amplificadores operacionais, apresentando um circuito amplificador somador, o qual é uma aplicação prática típica. A Figura 1 apresenta um amplificador somador com duas tensões de entrada, v_a e v_b e uma tensão de saída v_o , relacionadas conforme a equação (1).

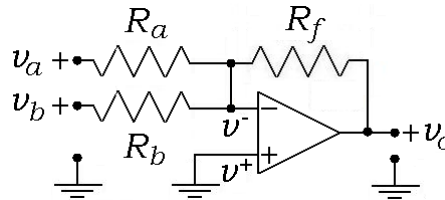


Figura 1. Circuito amplificador somador com duas entradas.

I.
$$v_o = -\left(\frac{R_f}{R_a} v_a + \frac{R_f}{R_b} v_b\right)$$

O amplificador operacional é um circuito eletrônico que se comporta como uma fonte de tensão controlada por tensão, em que

II.
$$v_o = A(v^+ - v^-)$$

Idealmente, o ganho a circuito aberto A é infinitamente grande, o que faz com que a tensão diferencial $v^+ - v^-$ seja nula em circuitos nos quais o amplificador operacional opere em sua região linear (não saturado). Também sob o ponto de vista ideal, a resistência na saída v_o é considerada nula, e a resistência entre as entradas v^+ e v^- **é infinitamente grande**.

Com base no exposto, conclui-se que as alternativas **I, II e IV são verdadeiras**. Já a alternativa **III** é falsa, pois um circuito seguidor de tensão tem a forma apresentada na Figura 2, na qual a tensão de saída v_o tem valor igual à tensão de entrada v_i .

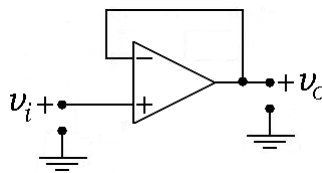


Figura 2. Circuito seguidor de tensão.

Finalmente, a alternativa **V** também é falsa, pois o valor da tensão de saída v_2 em relação ao ponto terra é dado por

$$v_2 = -\left(\frac{200}{1000} 6 + \frac{200}{1000} (6 + 6)\right) = -3,6 \text{ V.}$$

REFERÊNCIAS

ALEXANDER, Charles K. *Fundamentos de circuitos elétricos*. 3. ed. São Paulo: McGraw-Hill, 2008. 1014 p.

PERTENCE JÚNIOR, Antonio. *Amplificadores operacionais e filtros ativos: teoria, projetos, aplicações e laboratório*. 6. ed. Porto Alegre: Bookman, 2003. 304 p.

SEDRA, Adel S. *Microeletrônica*. 5. ed. São Paulo: Pearson, 2011. 848 p.

QUESTÃO 46

Em um modelo de dados que descreve a publicação acadêmica de pesquisadores de diferentes instituições em eventos acadêmicos, considere as tabelas abaixo.

DEPARTAMENTO (#CodDepartamento, NomeDepartamento)

EMPREGADO (#CodEmpregado, NomeEmpregado, CodDepartamento, Salario)

Na linguagem SQL, o comando mais simples para recuperar os códigos dos departamentos cuja média salarial seja maior que 2000 é

- A.

```
SELECT CodDepartamento
FROM EMPREGADO
GROUP BY CodDepartamento
HAVING AVG (Salario) > 2000
```
- B.

```
SELECT CodDepartamento
FROM EMPREGADO
WHERE AVG (Salario) > 2000
GROUP BY CodDepartamento
```
- C.

```
SELECT CodDepartamento
FROM EMPREGADO
WHERE AVG (Salario) > 2000
```
- D.

```
SELECT CodDepartamento, AVG (Salario) > 2000
FROM EMPREGADO
GROUP BY CodDepartamento
```
- E.

```
SELECT CodDepartamento
FROM EMPREGADO
GROUP BY CodDepartamento
ORDER BY AVG (Salario) > 2000
```

Resposta: alternativa (A)

Autor: Duncan Dubugras Alcoba Ruiz

COMENTÁRIO

Trata-se de consulta somente na tabela EMPREGADO, pois a mesma contém o código do departamento ao qual o empregado está vinculado. A média salarial pode ser obtida diretamente com a função de agregação AVG. Como a questão quer a média salarial por departamento, é necessário usar a cláusula GROUP BY com a coluna CodDepartamento. Ainda, como a condição de seleção é sobre o resultado de uma função de agregação, a média salarial seja maior que 2000, esta condição deve estar na cláusula HAVING. Logo, a alternativa correta é **A**.

A alternativa **A** é a única sintaticamente correta.

As alternativas **B** e **C** estão erradas porque, na cláusula WHERE, podem-se filtrar registros da tabela de entrada ANTES da agregação ser feita. A alternativa **D** coloca a condição de filtragem como saída do comando SELECT, enquanto a alternativa **E** a coloca na especificação da ordem na saída.

REFERÊNCIAS

ELMASRI, R.; NAVATHE, S. B. *Sistemas de Banco de Dados*. 4. ed. São Paulo: Pearson Brasil, 2005. 724p.

BOWMAN, J.; EMERSON, S.; DARNOVSKY, M. *The practical SQL handbook: using Structured Query Language*. 3. ed. Boston: Addison-Wesley, 1996. 454p.

DATE, C. J. *Introdução a sistemas de bancos de dados*. Rio de Janeiro: Campus, 2000. 454p.

HEUSER, C. *Projeto de banco de dados*. 5. ed. Porto Alegre: Sagra-DC Luzzatto, 2004. 236p.

PRICE, J. *Oracle Database 11g SQL*. Porto Alegre: Bookman, 2008. 684p.

RAMAKRISHNAN, R. *Database management systems*. Boston: McGraw-Hill, 2000. 906p.

SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN, S. *Sistema de bancos de dados*. 5. ed. Rio de Janeiro: Campus, 2006. 808p.

ULLMAN, J.; WIDOM, J. *A first course in database systems*. Nova Jersey: Prentice Hall, 1997. 470p.

QUESTÃO 47

Uma empresa de natureza estritamente operacional deseja implantar um setor de suporte ao processo de tomada de decisão, já que os resultados que vem apresentando demonstram contínua queda da margem de lucro e aumento do custo operacional. Para isso, os executivos de alto escalão da empresa decidiram investir na aquisição de uma ferramenta OLAP acoplada a uma data warehouse.

Nessa situação, avalie as afirmações a seguir.

- I. No que tange ao tipo de suporte propiciado, os sistemas OLAP podem ser classificados como sistemas de trabalhadores do conhecimento.
- II. Ferramentas OLAP apresentam foco orientado a assunto, em contraposição a sistemas OLTP, que são orientados a aplicação.
- III. Tendo em vista que data marts são construídos utilizando-se os sistemas legados da empresa, sem a utilização de dados externos, o processo de extração, transformação e carga envolve a integração de dados, suprimindo-se a tarefa de limpeza.
- IV. O projeto de um data warehouse define a forma com que a base de dados será construída. Uma das opções é a abordagem data mart, em que os diversos data marts são integrados, até que se obtenha, ao final do processo, um data warehouse da empresa.

É correto o que se afirma em

- A. I e III, apenas.
- B. I e IV, apenas.
- C. II e III, apenas.
- D. II e IV, apenas.
- E. I, II, III e IV.

Resposta: alternativa (D)

Autor: Duncan Dubugras Alcoba Ruiz

COMENTÁRIO

Trata-se de questão sobre processamento analítico de dados e armazéns de dados. Nesse contexto, quer se saber quais das quatro afirmações da questão estão corretas.

A afirmação **I** está incorreta. Um sistema OLAP é um SAD – Sistema de Apoio a Decisão.

A afirmação **II** está correta. Para Inmon (1997), “um data warehouse é um conjunto de dados baseado em assuntos, integrado, não volátil e variável em relação ao tempo, de apoio às decisões gerenciais”.

A afirmação **III** está incorreta. A limpeza dos dados sempre é necessária para tentar completar dados faltantes, amenizar a presença de ruídos, identificar *outliers*, e corrigir inconsistências nos dados, mesmo que os dados externos terminem.

A afirmação **IV** está correta. É uma das opções apresentadas por Kimball (2002) e por Inmon (1997) para a construção de um armazém de dados.

REFERÊNCIAS

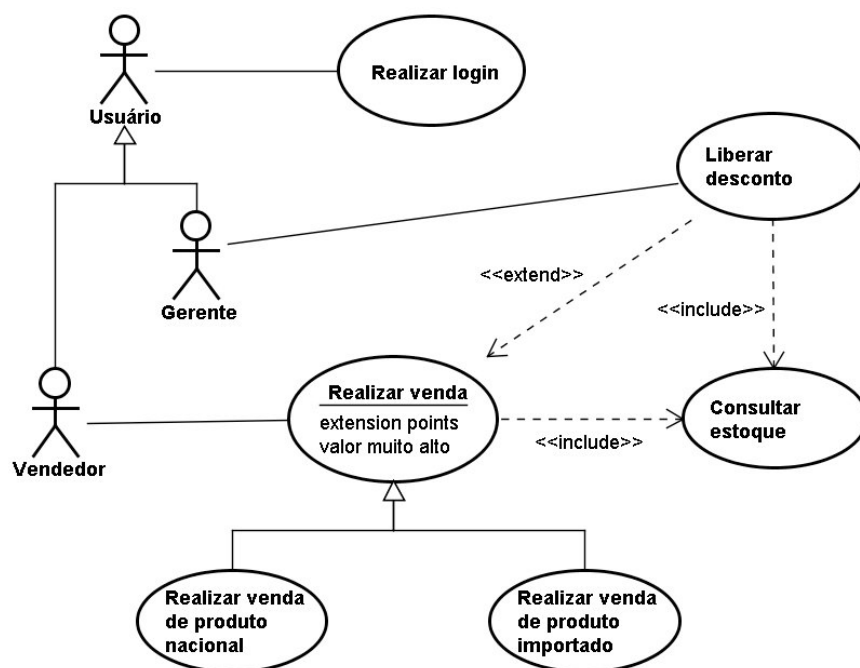
INMON, W. H. *Como construir o Data Warehouse*. Rio de Janeiro: Editora Campus, 1997. 388p.

LAUDON, Kenneth C. *Sistemas de informação gerenciais*. 7. ed. São Paulo: Pearson, 2010. 452 p.

KIMBALL, R. *The data warehouse toolkit: the complete guide to dimensional modeling*. 2. ed. Nova York: John Wiley & Sons, 2002. 436 p.

QUESTÃO 48

No desenvolvimento de um software para um sistema de venda de produtos nacionais e importados, o analista gerou o diagrama de casos de uso a seguir.



Da análise do diagrama, conclui-se que

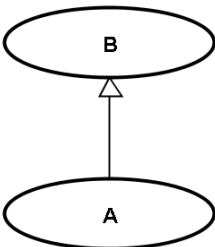

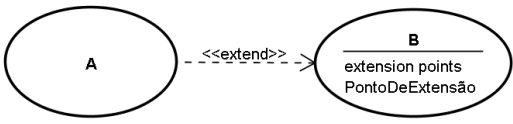
- A. a execução do caso de uso 'Consultar estoque' incorpora opcionalmente o caso de uso 'Liberar desconto'.
- B. a execução do caso de uso 'Liberar desconto' incorpora opcionalmente o caso de uso 'Realizar venda'.
- C. a execução do caso de uso 'Realizar venda' incorpora obrigatoriamente o caso de uso 'Consultar estoque'.
- D. a execução do caso de uso 'Realizar venda de produto nacional' incorpora obrigatoriamente o caso de uso 'Liberar desconto'.
- E. um Gerente pode interagir com o caso de uso 'Realizar venda', pois ele é um Usuário.

Resposta: alternativa (C)

Autor: Marcelo Hideki Yamaguti

COMENTÁRIO

A questão aborda a interpretação de um diagrama de Casos de Uso que envolve atores, casos de uso e relacionamentos. Para a resolução da questão, é importante o conhecimento do significado dos relacionamentos:

RELACIONAMENTO	DIAGRAMA	SIGNIFICADO
Generalização		Relacionamento entre casos de uso (ou atores) que indica que um caso de uso (ou ator) A herda as características do caso de uso (ou ator) B.
<<include>>		Relacionamento estereotipado entre casos de uso que indica que um caso de uso A incorpora obrigatoriamente outro caso de uso B.
<<extend>>		Relacionamento estereotipado entre casos de uso que indica que um caso de uso A é incorporado opcionalmente pelo caso de uso B na ocorrência do ponto de extensão.

Dentre as opções apresentadas, a única correta é a alternativa C (“a execução do caso de uso ‘Realizar venda’ incorpora obrigatoriamente o caso de uso ‘Consultar estoque’”).

Quanto às demais opções:

- Na alternativa **A**, o correto seria “a execução do caso de uso ‘Liberar desconto’ incorpora obrigatoriamente o caso de uso ‘Consultar estoque’”.
- Na alternativa **B**, o correto seria “a execução do caso de uso ‘Realizar venda’ incorpora opcionalmente o caso de uso ‘Liberar desconto’”.
- Na alternativa **D**, o correto seria “a execução do caso de uso ‘Realizar venda de produto nacional’ incorpora opcionalmente o caso de uso ‘Liberar desconto’”.
- Na alternativa **E**, o correto seria “um Gerente pode interagir com o caso de uso ‘Realizar login’, pois ele é um Usuário”.

REFERÊNCIAS

ARLOW, J.; NEUSTADT, I. UML 2 and the Unified Process: practical object-oriented analysis and design. 2. ed. Upper Saddle River: Addison Wesley, 2005. 592p.

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. UML: guia do usuário. 2. ed. Rio de Janeiro: Elsevier, 2006. 474p.

FOWLER, M. UML essencial: um breve guia para a linguagem-padrão de modelagem de objetos. 2. ed. Porto Alegre: Bookman, 2000. 169p.

JACOBSON, I.; BOOCH, G.; RUMBAUGH, J. The unified modeling language reference manual. Reading: Addison-Wesley, c1999. 550p.

LARMAN, C. Utilizando UML e padrões: uma introdução à análise e projeto orientados a objetos. 3. ed. São Paulo: Bookman, 2007. 696p.

QUESTÃO 49

Considerando o conceito de sistema, trazido pela Teoria Geral de Sistemas, um projeto de desenvolvimento de software poderia ser considerado como um sistema aberto.

Nessa perspectiva, solicitações de mudanças originadas de um stakeholder externo e que afetam o projeto podem ser consideradas como

- A. ambiente.
- B. entrada.
- C. feedback.
- D. processos.
- E. saída.

Resposta: alternativa (B)

Autor: Gilberto Keller de Andrade

COMENTÁRIO

A questão inicia com uma afirmação verdadeira: a Teoria Geral de Sistemas considera que todo sistema aberto (uma comunidade de entidades conectadas com a finalidade de alcançar um ou mais objetivos operando dentro de um ambiente) sofre influência do meio em que “atua” ou o influencia. Nesse contexto, um projeto de desenvolvimento de software pode ser visto como um sistema, pois, em um projeto, nós temos pessoas, equipamentos, regras, organização, estruturas, todas conectadas, formando um todo que se espera que seja organizado, na busca de objetivos: o software. Esse sistema é aberto, pois foi criado para atender necessidades de algum cliente, usar recursos de fornecedores diversos, produzir um software que influenciará outras entidades que estão em seu ambiente. A partir de 1948, com o surgimento da Cibernética, o conceito de feedback, como um processo de realimentação e controle, é incorporado no modelo geral de um sistema aberto, pois este mecanismo (feedback) ajuda o sistema a não se desviar de seus propósitos por falta de controle. Assim, podemos resumir que existem cinco elementos mínimos, necessários à especificação de um sistema aberto: entradas (provenientes do ambiente), saídas (resultados gerados pelo sistema para o ambiente), transformações (responsável

pelas mudanças das entradas em saídas), ambiente (contexto do sistema) e feedback (responsável pelo controle, uma espécie de administração do sistema, na visão de C. W. Churchmann (Churchmann, 1971)).

Assim, uma “solicitação de mudança” feita por um agente externo (*stakeholder*) só pode ser uma entrada (alternativa **B**), uma vez que “a solicitação” não é um processo, nem de mudança, nem de *feedback*, não é uma saída, não é ambiente, mesmo que tenha sido gerada no ambiente.

REFERÊNCIA

CHURCHMAN, C. W. *Introdução à Teoria Geral dos Sistemas*. São Paulo: Vozes, 1971.

QUESTÃO 50

Uma empresa vem desenvolvendo um programa de melhoria de seus processos de software utilizando o modelo de qualidade CMMI. O programa envolveu a definição de todos os processos padrão da organização, implementação de técnicas de controle estatístico de processos e métodos de melhoria contínua. Após a avaliação SCAMPI, classe A, foi detectado que a área de processo de PP - Project Planning (Planejamento de Projeto) não estava aderente ao modelo.

Nesse contexto, considerando a representação por estágios do CMMI, a empresa seria classificada em que nível de maturidade?

- A. Nível 1.
- B. Nível 2.
- C. Nível 3.
- D. Nível 4.
- E. Nível 5.

Resposta: alternativa (A)

Autora: Sabrina dos Santos Marczak

COMENTÁRIO

O modelo CMMI (*Capability Maturity Model Integration*) é um modelo que guia o desenvolvimento ou avaliação da maturidade dos processos de software de um projeto, um departamento ou uma organização como um todo. A maturidade do processo indica a capacidade do mesmo de fornecer uma previsibilidade dos seus resultados. Ou seja, indica quanto os resultados gerados por um processo de software poderão ser alcançados com a execução dos mesmos.

Visando a uma progressão gradativa da maturidade dos processos relacionados ao ciclo de desenvolvimento de software, o modelo CMMI organizou o mesmo em níveis de maturidade. Os níveis de maturidade são os seguintes: nível 1 – inicial, os processos são imprevisíveis e reativos às ações dos indivíduos; nível 2 – gerenciado, os processos são definidos por projetos e ainda são, em geral, reativos; nível 3 – definido, os processos são padronizados em nível organizacional e decisões são feitas *a priori*

quanto à customização dos processos (quando necessário), justificando as mudanças realizadas; nível 4 – quantitativamente gerenciados, os processos são medidos e controlados; e nível 5 – em otimização, os processos focam na busca de uma melhoria contínua, otimizando seus recursos e atividades.

Cada nível de maturidade é composto de áreas de processo. Uma área de processo reúne as práticas esperadas para um determinado fim. Por exemplo, a área de processo denominada “Gerência de Requisitos” reúne práticas que visam gerenciar os requisitos dos produtos do projeto, identificando inconsistências entre os requisitos, o plano de projeto e os produtos de trabalho gerados pelo projeto.

Para que um nível de maturidade seja considerado como aderente ao modelo, todas as áreas de processo precisam ter suas práticas definidas e institucionalizadas no projeto, departamento ou organização. Caso uma das áreas de processo não esteja aderente, o projeto, departamento ou organização será então considerado como um nível de maturidade inferior ao nível a que a área de processo está relacionada.

A área de processo PP – *Project Planning* (Planejamento de Projeto) – é uma área do nível 2 de maturidade. Dessa forma, se esta área não está aderente ao modelo, então a empresa será classificada com um nível de maturidade 1 – inicial. Sendo assim, a resposta correta é a alternativa A.

REFERÊNCIA

CHRISSIS, M. B.; KONRAD, M.; SHRUM, S. *CMMI: Guidelines for Process Integration and Product Improvement*. 2. ed. Nova York: Addison-Wesley, 2006. 704p.

QUESTÃO DISCURSIVA 3

Os números de Fibonacci correspondem à uma sequência infinita na qual os dois primeiros termos são 0 e 1. Cada termo da sequência, à exceção dos dois primeiros, é igual à soma dos dois anteriores, conforme a relação de recorrência abaixo.

$$f_n = f_{n-1} + f_{n-2}$$

Desenvolva dois algoritmos, um iterativo e outro recursivo, que, dado um número natural $n > 0$, retorna o n -ésimo termo da sequência de Fibonacci. Apresente as vantagens e desvantagens de cada algoritmo. (valor: 10,0 pontos)

Autora: Lucia Maria Martins Giraffa

COMENTÁRIO

A sequência de números naturais, denominada de Fibonacci, associada à recorrência $f_n = f_{n-1} + f_{n-2}$ é expressa por:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, ...

Um número integrante desta série é obtido, a partir do terceiro, pela soma dos seus dois antecessores. Por definição os dois primeiros termos da série são 0 e 1. Isso é importante para a elaboração do algoritmo. Destaca-se que o enunciado pede o n -ésimo número apenas a ser impresso e não a série inteira até aquele número. Será necessário gerar todos os valores até o n -ésimo e imprimir apenas este.

Este algoritmo, quando escrito da forma iterativa, é implementado diretamente da definição da série e pode ser construído usando-se as opções existentes para comandos relacionados às estruturas de repetição. A opção com Para (*for*) é a mais usada.

A versão iterativa tem complexidade linear, o que a torna mais vantajosa em termos de eficiência, mas exige mais atenção na implementação. Importante na versão iterativa é que apareça de forma explícita a inicialização das variáveis auxiliares que representam os dois primeiros termos, os testes

para saída direta desses dois primeiros termos e que a estrutura de repetição utilize um acumulador e as atribuições que alteram os valores, seguindo esta lógica:

```
aux ← valor_anterior+ valor_corrente
valor_anterior ← valor_corrente
valor_corrente ← aux
```

Uma opção iterativa de algoritmo seria:

```
a←0
b←1
se (termo←1) escrever a
se (termo←2) escrever b
se (termo >=3)
  início
    para int i=3 até i <=termo incremento de 1
      início
        soma←a+b
        a←b
        b←soma
      fim
    escrever soma
  fim
```

Outra opção iterativa seria:

```
se n = 0 ou n = 1 então retorna 1
senão
  início
    a ← 0
    b ← 1
    para i ← 1 até n - 1 incremento 1
      início
        aux ← a + b
        a ← b
        b ← aux
      fim
    retorna aux
  fim
```

Os algoritmos recursivos quase sempre consomem mais recursos (especialmente memória) do computador, logo, tendem a apresentar um desempenho inferior aos iterativos.

No caso dessa **Série**, o algoritmo proposto deve considerar os seguintes componentes:

```
...int ... Fib(int n)

início
se n = 0 ou n = 1 então
  retorna n
senão
  retorna Fib(n-2) + Fib(n-1)
fim
```

Ou uma opção desse poderia ser assim:

```
...int ... Fib(int n)

início
se n = 1 retorna 0
se n = 2 retorna 1
senão
  retorna Fib(n-2) + Fib(n-1)
fim
```

Essa implementação poderá ser uma função ou um método dentro de uma classe dependendo do paradigma que o programador está usando. Razão pela qual usamos os “...” (três pontos) para destacar isso e chamamos atenção nos exemplos relacionados às opções recursivas nos exemplos acima.

REFERÊNCIAS

FORBELLONE, A. L. V.; EBERSPACHER, H. F. *Lógica de Programação*. São Paulo: Makron Books, 2005.

GOODRICH, M. T.; TAMASSIA, R. *Estruturas de Dados e Algoritmos em Java*. 4a ed., Porto Alegre, Bookman, 2007.

HORSTMANN, C. *Big Java*. Porto Alegre: Bookman, 2004.

SZWARCFITER, J. L.; MARKENZON, L. *Estruturas de Dados e seus Algoritmos*. 2. ed. Rio de Janeiro. LTC Editora, 1997.

QUESTÃO DISCURSIVA 4

Listas ordenadas implementadas com vetores são estruturas de dados adequadas para a busca binária, mas possuem o inconveniente de exigirem custo computacional de ordem linear para a inserção de novos elementos. Se as operações de inserção ou remoção de elementos forem frequentes, uma alternativa é transformar a lista em uma árvore binária de pesquisa balanceada, que permitirá a execução dessas operações com custo logarítmico.

Considerando essas informações, escreva um algoritmo recursivo que construa uma árvore binária de pesquisa completa, implementada por estruturas auto referenciadas ou apontadores, a partir de um vetor ordenado, v , de n inteiros, em que $n = 2^m - 1$, $m > 0$. O algoritmo deve construir a árvore em tempo linear, sem precisar fazer qualquer comparação entre os elementos do vetor, uma vez que este já está ordenado. Para isso,

- A. descreva a estrutura de dados utilizada para a implementação da árvore (valor = 2,0 pontos)
- B. escreva o algoritmo para a construção da árvore. A chamada principal à função recursiva deve passar, como parâmetros, o vetor, índice do primeiro e último elementos, retornando a referência ou apontador para a raiz da árvore criada (valor: 8,0 pontos).

Observação: Qualquer notação em português estruturado, de forma imperativa ou orientada a objetos deve ser considerada, assim como em uma linguagem de alto nível, como o Pascal, C e Java.

Autor: Marcelo Cohen

COMENTÁRIO

A questão engloba a transformação de uma estrutura de dados linear (vetor) em uma hierárquica (árvore binária balanceada), com o objetivo específico de melhorar o desempenho das operações de inserção e remoção de elementos. O problema da inserção em um vetor é que os elementos sempre precisam ser deslocados na memória, o que exige um custo computacional linear. Além disso, a inserção de elementos em um vetor requer que esse tenha espaço livre: se não for o caso, ainda haverá a necessidade de alocar um vetor maior e copiar os elementos do original para ele. Isso aumenta ainda mais o custo da operação. Já numa estrutura de árvore, a inserção é trivial: basta localizar o nodo cor-

reto (o que tem um custo logarítmico, como afirma o enunciado) e inserir o novo elemento à direita ou à esquerda, dependendo da relação de ordem entre o valor dele e do nodo localizado.

Para realizar a transformação, a questão solicita explicitamente um algoritmo de complexidade linear, ou seja, cujo tempo seja diretamente proporcional à quantidade de itens no vetor original.

Resposta e comentário do item A. Para a criação de uma árvore binária de pesquisa, é necessária a representação de um **nodo** dessa árvore. Uma estrutura possível na linguagem C++ seria um *template* de classe com ponteiros. Dessa forma, será possível armazenar qualquer tipo de dado ordenável no nodo, isto é, que possa ser comparado usando os operadores relacionais (<, >, <=, >=, == ou !=).

Qualquer linguagem de programação poderia ser utilizada desde que a resposta contivesse os elementos sublinhados: o valor armazenado, um ponteiro/referência para o *Nodo* à direita e outro ponteiro/referência para o *Nodo* à esquerda.

```
template<typename T>
class Nodo {
    T valor;
    Nodo* esq;
    Nodo* dir;

public:

    Nodo(T valor) {
        this->valor = valor;
        this->esq = this->dir = NULL;
    }

    T getValor() { return valor; }
    Nodo* getEsq() { return esq; }
    Nodo* getDir() { return dir; }

    void setDir(Nodo* n) { dir = n; }
    void setEsq(Nodo* n) { esq = n; }
};
```

Resposta e comentário do item B. Algoritmo recursivo, implementado como uma função genérica (*template*) em C++:

```
template<typename T>
Nodo<T>* gera_abp(T lista[], int primeiro, int ultimo)
{
    if (primeiro > ultimo)
        return NULL;

    int meio = (primeiro+ultimo)/2;

    Nodo<T>* raiz = new Nodo<T>(lista[meio]);
    raiz->setEsq(gera_abp(lista, primeiro, meio-1));
    raiz->setDir(gera_abp(lista, meio+1, ultimo));

    return raiz;
}
```

Para se criar uma árvore binária de pesquisa a partir do vetor, é preciso escolher um dos elementos para ser a raiz. Como o enunciado solicita uma árvore **balanceada**, a melhor escolha é o elemento exatamente no **meio** do vetor. A partir daí, a árvore é gerada recursivamente, considerando-se a metade esquerda e direita. Em outras palavras, o processo é repetido, novamente escolhendo-se o elemento no meio de cada partição. Isso garante o balanceamento da árvore e caracteriza, como solicita o enunciado, um algoritmo de complexidade linear, uma vez que cada elemento do vetor só será visitado uma única vez.

REFERÊNCIAS

GOODRICH, Michael T. *Estruturas de dados e algoritmos em Java*. 4. ed. Porto Alegre: Bookman, 2007. 600 p.

MCALLISTER, W. *Data Structures and Algorithms Using Java*. 1. ed. Boston: Jones and Bartlett, 2009. 580 p.

QUESTÃO DISCURSIVA 5

As memórias cache são usadas para diminuir o tempo de acesso à memória principal, mantendo cópias de seus dados. Uma função de mapeamento é usada para determinar em que parte da memória cache um dado da memória principal será mapeado. Em certos casos, é necessário usar um algoritmo de substituição para determinar qual parte da cache será substituída.

Suponha uma arquitetura hipotética com as seguintes características:

- A memória principal possui 4 Gbytes, em que cada byte é diretamente endereçável com um endereço 32 bits.
- A memória cache possui 512 Kbytes, organizados em 128 K linhas de 4 bytes.
- Os dados são transferidos entre as duas memórias em blocos de 4 bytes.

Considerando os mapeamentos direto, totalmente associativo e associativo por conjuntos (em 4 vias), redija um texto que contemple as organizações dessas memórias, demonstrando como são calculados os endereços das palavras, linhas (blocos), rótulos (tags) e conjunto na memória cache em cada um dos três casos. Cite as vantagens e desvantagens de cada função de mapeamento, bem como a necessidade de algoritmos de substituição em cada uma delas. (valor: 10,0 pontos)

Autor: Alexandre de Moraes Amory

COMENTÁRIO

O crescimento de desempenho dos processadores é tipicamente muito mais acentuado que o crescimento de desempenho das memórias, efeito chamado de *Memory Gap*. Isso faz com que, a longo prazo, o custo de tempo de acesso à memória se torne cada vez mais relevante para o desempenho de um programa. Dessa forma, é importante que os alunos estejam cientes do funcionamento da hierarquia de memória para aplicações com restrições de desempenho.

Dado que a especificação de hierarquia de memória é apresentada e assumindo que a memória em questão é endereçada em palavras de 1 byte, essa pode ser configurada das seguintes formas:

- Mapeamento direto: 2 bits para endereçar o byte do bloco, 17 para endereçamento da linha da cache, e 13 bits de rótulo.
- Mapeamento associativo: 2 bits para endereçar o byte do bloco e 30 bits de rótulo.
- Mapeamento conjunto associativo: 2 bits para endereçar o byte do bloco, 15 bits para endereçar o conjunto (128 K linhas/4 vias), e 15 bits de rótulo.

O mapeamento direto é o que tem implementação de hardware mais simples e barata, porém possui a tendência de gerar mais erros de *cache*, pois os endereços de memória são mapeados em endereços fixos da *cache*, aumentando o número de conflitos na *cache*. O mapeamento associativo possui o menor número de conflitos de *cache*, pois um endereço da memória pode ser associado a qualquer linha da memória *cache*. Por outro lado, o custo de implementação do hardware é alto e não escalável, ou seja, o custo aumenta rapidamente à medida que o tamanho da *cache* também aumenta. O mapeamento conjunto associativo é um compromisso entre os dois métodos anteriores. Ele não é tão restritivo quanto o mapeamento direto, reduzindo o número de conflitos na *cache*, porém, o custo de hardware não é tão alto quanto o mapeamento direto. Experimentos demonstram que aumentar a associatividade demais não traz necessariamente benefícios de desempenho. Por esse motivo o mapeamento conjunto associativo é o mais utilizado nos processadores reais, pois provê baixo nível de conflito e custo de hardware aceitável.

Os algoritmos de substituição são utilizados quando um novo bloco precisa ser escrito em uma memória *cache* com suas linhas já preenchidas. Dessa forma, uma linha deve ser selecionada para receber o novo bloco de dados. Um exemplo de algoritmos de substituição é chamado de *Least Recently Used* (LRU), que seleciona a linha usada por mais tempo para ser substituída. Os algoritmos de substituição só fazem sentido nos métodos de mapeamento nos quais um determinado endereço da memória possa ser mapeado em múltiplas linhas da *cache*. Dessa forma, o mapeamento direto não necessita de algoritmo de substituição.

REFERÊNCIAS

MONTEIRO, M. A. *Introdução à Organização de Computadores*. 5. ed. Rio de Janeiro: LTC, 2012.

PATTERSON, David; HENESSY, John L. *Arquitetura de computadores: uma abordagem quantitativa*. 4. ed. Rio de Janeiro: Campus, 2008.

_____; _____. *Organização e projeto de computadores: a interface hardware/software*. 4. ed. Rio de Janeiro: Elsevier, 2014.

STALLINGS, W. *Arquitetura e Organização de Computadores*. 8. ed. São Paulo: Pearson, 2010.

TANENBAUM, A. S. *Organização Estruturada de Computadores*. 5. ed. São Paulo: Pearson, 2007.